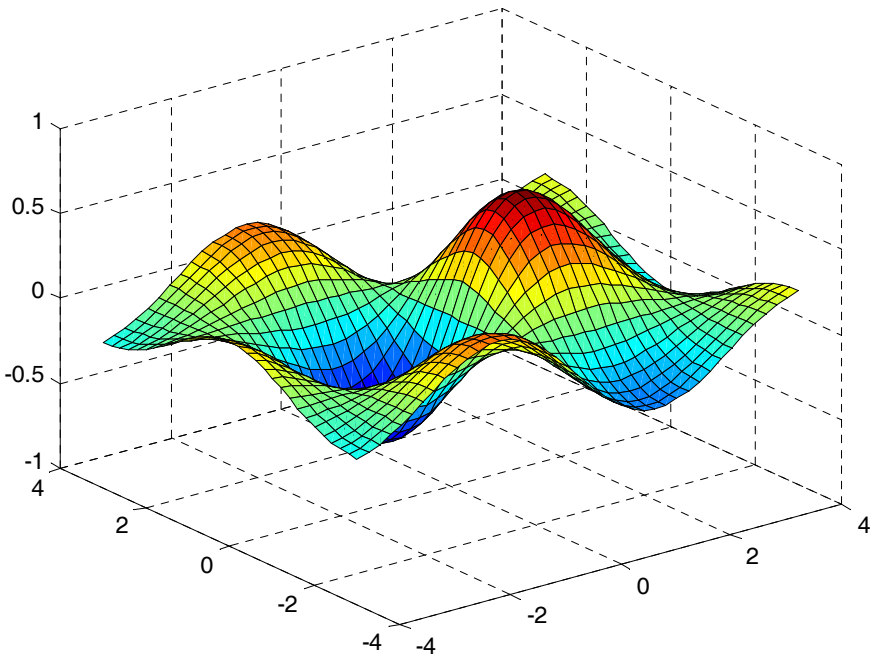
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΕΙΣΑΓΩΓΗ ΣΤΟ MATLAB – SIMULINK



ΑΡΝΑΟΥΤΑΚΗΣ ΝΕΚΤΑΡΙΟΣ

ΝΟΕΜΒΡΙΟΣ 2005

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

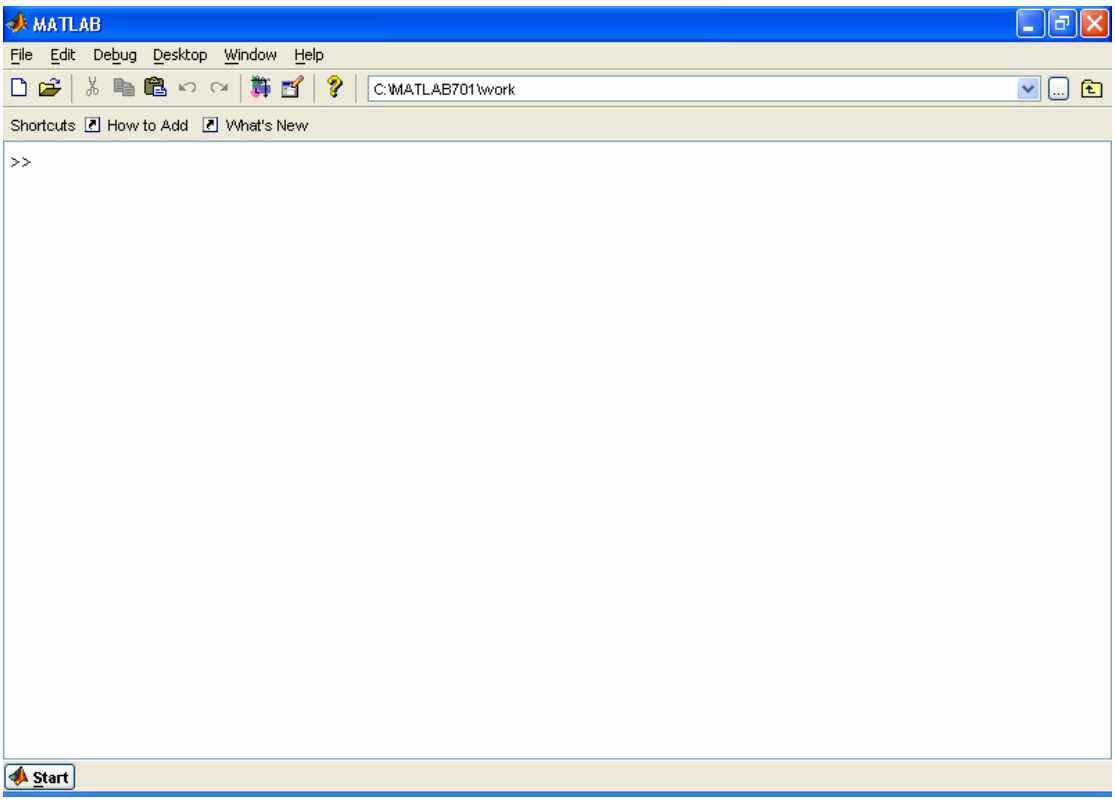
ΚΕΦΑΛΑΙΟ 1
ΠΕΡΙΓΡΑΦΗ ΚΑΙ ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ MATLAB (ΕΚΔΟΣΕΙΣ 7.X)

1.1. Γενικά

Η γλώσσα προγραμματισμού MATLAB (το όνομα προήλθε από τις λέξεις Matrix Laboratory) λειτουργεί ως διερμηνέας εντολών (command interpreter), οι οποίες δίνονται μέσω του παραθύρου εντολών της (MATLAB command window). Οι εντολές αυτές μπορεί να είναι:

1. ορισμοί μεταβλητών και πράξεις
2. κλήση ενσωματωμένων συναρτήσεων της MATLAB και των εγκατεστημένων εργαλειοθηκών της (toolboxes)
3. κλήση συναρτήσεων (functions) ή αρχείων εντολών MATLAB (scripts) που κατασκευάζονται από τους χρήστες με τη μορφή m-file

Το περιβάλλον εργασίας της MATLAB απεικονίζεται παρακάτω (Εικόνα 1.1):



Εικόνα 1.1

Η ευρεία χρήση της MATLAB οφείλεται σε μεγάλο βαθμό στην επεκτασιμότητα της μέσω των διάφορων εργαλειοθηκών, κάθε μια από τις οποίες περιέχει ένα αριθμό συναρτήσεων για ένα συγκεκριμένο αντικείμενο. Η δομή των υπαρχόντων στοιχείων σε μια εγκατάσταση MATLAB παρουσιάζεται εκτελώντας την εντολή *help* (γράφουμε help δίπλα από την προτροπή » και πατάμε enter):

```
>> help
HELP topics

matlab\general       - General purpose commands.
matlab\ops           - Operators and special characters.
matlab\lang          - Programming language constructs.
matlab\elmat         - Elementary matrices and matrix manipulation.
matlab\elfun         - Elementary math functions.
matlab\specfun       - Specialized math functions.
matlab\matfun        - Matrix functions - numerical linear algebra.
matlab\datafun       - Data analysis and Fourier transforms.
matlab\polyfun       - Interpolation and polynomials.
matlab\funfun        - Function functions and ODE solvers.
matlab\sparfun       - Sparse matrices.
matlab\scribe        - Annotation and Plot Editing.
matlab\graph2d       - Two dimensional graphs.
matlab\graph3d       - Three dimensional graphs.
matlab\specgraph     - Specialized graphs.
matlab\graphics      - Handle Graphics.
matlab\uitools       - Graphical user interface tools.
matlab\strfun        - Character strings.
matlab\imagesci      - Image and scientific data input/output.
matlab\iofun         - File input and output.
matlab\audiovideo    - Audio and Video support.
matlab\timefun       - Time and dates.
matlab\datatypes     - Data types and structures.
matlab\verctrl       - Version control.
matlab\codetools     - Commands for creating and debugging code.
matlab\helptools     - Help commands.
matlab\winfun        - Windows Operating System Interface Files (COM/DDE)
matlab\demos         - Examples and demonstrations.
toolbox\local        - Preferences.
matlabxl\matlabxl    - MATLAB Builder for Excel
simulink\simulink    - Simulink
simulink\blocks      - Simulink block library.
simulink\components  - Simulink components.
simulink\fixedandfloat - Simulink Fixed Point utilities.
fixedandfloat\fxpdemos - Fixed-Point Blockset Demos
fixedandfloat\obsolete - (No table of contents file)
simulink\simdemos    - Simulink 4 demonstrations and samples.
simdemos\aerospace   - Simulink: Aerospace model demonstrations and samples.
simdemos\automotive  - Simulink: Automotive model demonstrations and samples.
simdemos\simfeatures - Simulink: Feature demonstrations and samples.
simfeatures\mdlref   - (No table of contents file)
simdemos\simgeneral  - Simulink: General model demonstrations and samples.
simdemos\simnew      - Simulink: New features model demonstrations and samples.
simulink\dee         - Differential Equation Editor
shared\dastudio      - (No table of contents file)
stateflow\stateflow  - Stateflow
rtw\rtw              - Real-Time Workshop
shared\hds           - (No table of contents file)
```

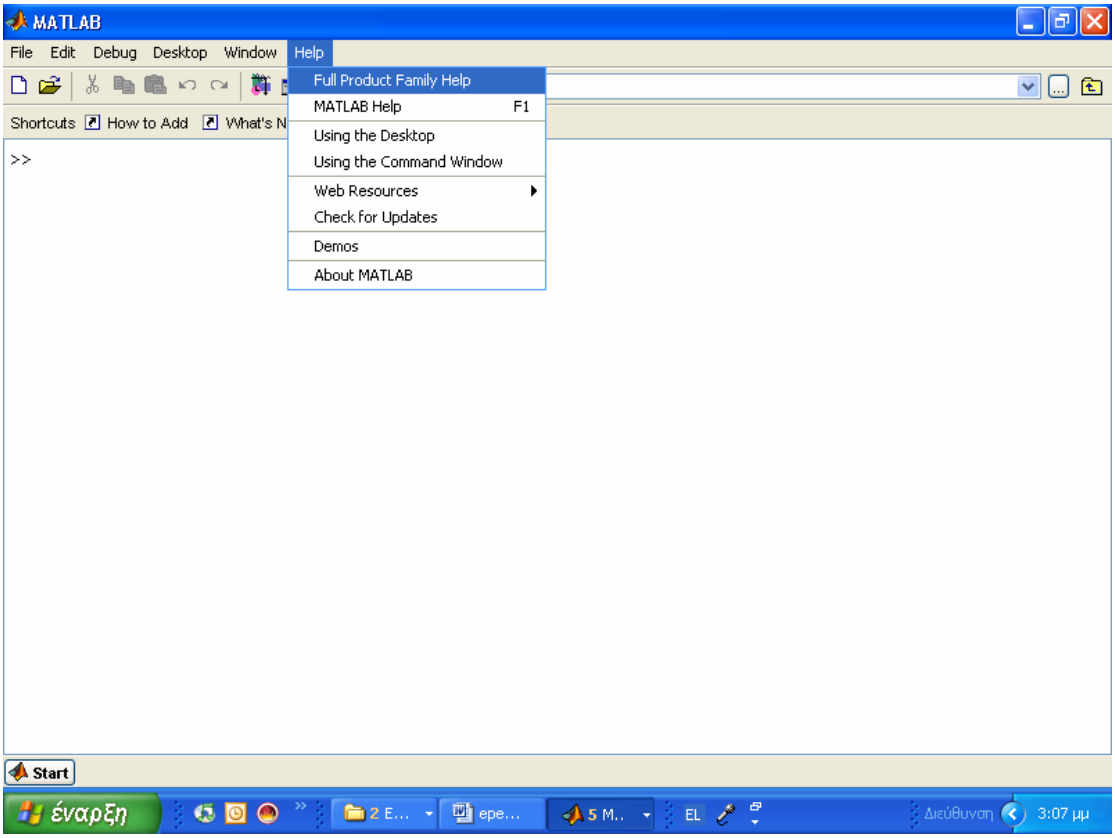
shared\timeseries - Shared Time Series Toolbox Library
stateflow\sfdemos - Stateflow demonstrations and samples.
stateflow\coder - Stateflow Coder
rtw\rtwdemos - Real-Time Workshop Demos
rtwdemos\rsimdemos - (No table of contents file)
asap2\asap2 - (No table of contents file)
asap2\user - (No table of contents file)
rtwin\rtwin - Real-Time Windows Target
simulink\accelerator - Simulink Accelerator
rtw\accel - (No table of contents file)
aeroblks\aeroblks - Aerospace Blockset
aeroblks\aerodemos - Aerospace Blockset demonstrations and examples.
aerodemos\texture - (No table of contents file)
bioinfo\bioinfo - Bioinformatics Toolbox
bioinfo\microarray - Bioinformatics Toolbox -- Microarray support functions.
bioinfo\proteins - Bioinformatics Toolbox -- Protein analysis tools.
bioinfo\biomatrices - Bioinformatics Toolbox -- Sequence similarity scoring matrices.
bioinfo\biodemos - Bioinformatics Toolbox -- Tutorials, demos and examples.
can\blocks - (No table of contents file)
configuration\resource - (No table of contents file)
common\tgtcommon - (No table of contents file)
c166\c166 - Embedded Target for Infineon C166 Microcontrollers
c166\blocks - (No table of contents file)
c166\c166demos - (No table of contents file)
ccslink\ccslink - Link for Code Composer Studio
ccslink\ccsblks - (No table of contents file)
ccslink\ccsdemos - Link for Code Composer Studio® Demos
cdma\cdma - CDMA Reference Blockset
cdma\cdmamasks - CDMA Reference Blockset mask helper functions.
cdma\cdmamex - CDMA Reference Blockset S-Functions.
cdma\cdmademos - CDMA Reference Blockset demonstrations and examples.
combuilder\combuilder - MATLAB Builder for COM
comm\comm - Communications Toolbox
comm\commdemos - Communications Toolbox Demonstrations.
commdemos\commdocdemos - Communications Toolbox Documentation Examples.
comm\commobsolete - Archived MATLAB Files from Communications Toolbox Version 1.5.
commblks\commblks - Communications Blockset
commblks\commmasks - Communications Blockset mask helper functions.
commblks\commmex - Communications Blockset S-functions.
commblks\commblksdemos - Communications Blockset Demos.
commblksobsolete\v2p5 - (No table of contents file)
commblksobsolete\v2 - (No table of contents file)
commblksobsolete\v1p5 - Archived Simulink Files from Communications Toolbox Version 1.5.
toolbox\compiler - MATLAB Compiler
control\control - Control System Toolbox
control\ctrlguis - Control System Toolbox -- GUI support functions.
control\ctrlobsolete - Control System Toolbox -- obsolete commands.
control\ctrlutil - (No table of contents file)
control\ctrldemos - Control System Toolbox -- Demos.
shared\controllib - Control Library
curvefit\curvefit - Curve Fitting Toolbox
curvefit\cftoolgui - (No table of contents file)
shared\optimlib - Optimization Library
daq\daq - Data Acquisition Toolbox
daq\daqguis - Data Acquisition Toolbox - Data Acquisition Soft Instruments.
daq\daqdemos - Data Acquisition Toolbox - Data Acquisition Demos.
database\database - Database Toolbox
database\dbdemos - Database Toolbox Demonstration Functions.
database\vqb - Visual Query Builder functions.
datafeed\datafeed - Datafeed Toolbox

datafeed\dfgui - Datafeed Toolbox Graphical User Interface
dspblks\dspblks - Signal Processing Blockset
dspblks\dspmasks - Signal Processing Blockset mask helper functions.
dspblks\dspmex - DSP Blockset S-Function MEX-files.
dspblks\dspdemos - Signal Processing Blockset demonstrations and examples.
targets\ecoder - Real-Time Workshop Embedded Coder
ecoder\ecoderdemos - (No table of contents file)
targets\mpt - (No table of contents file)
mpt\mpt - Module Packaging Tool
mpt\user_specific - (No table of contents file)
toolbox\exlink - Excel Link
symbolic\extended - Extended Symbolic Math
filterdesign\filterdesign - Filter Design Toolbox
filterdesign\quantization - (No table of contents file)
filterdesign\filtdesdemos - Filter Design Toolbox Demonstrations.
finance\finance - Financial Toolbox
finance\calendar - Financial Toolbox calendar functions.
finance\findemos - Financial Toolbox demonstration functions.
finance\finsupport - (No table of contents file)
finderiv\finderiv - Financial Derivatives Toolbox
finfixed\finfixed - Fixed-Income Toolbox
fixedpoint\fixedpoint - Fixed-Point Toolbox
fixedpoint\fidemos - (No table of contents file)
fixedpoint\fimex - (No table of contents file)
toolbox\fixpoint - Simulink Fixed Point
ftseries\ftseries - Financial Time Series Toolbox
ftseries\ftsdemos - (No table of contents file)
ftseries\ftsdata - (No table of contents file)
ftseries\ftstutorials - (No table of contents file)
fuzzy\fuzzy - Fuzzy Logic Toolbox
fuzzy\fuzdemos - Fuzzy Logic Toolbox Demos.
toolbox\gads - (No table of contents file)
gads\gads - Genetic Algorithm Direct Search Toolbox
gads\gadsdemos - Genetic Algorithm Direct Search Toolbox
garch\garch - GARCH Toolbox
garch\garchdemos - (No table of contents file)
toolbox\gauges - Gauges Blockset
hc12\hc12 - Embedded Target for Motorola HC12
hc12\blocks - (No table of contents file)
hc12\codewarrior - (No table of contents file)
hc12\hc12demos - (No table of contents file)
hdlfilter\hdlfilter - Filter Design HDL Coder
hdlfilter\hdlfiltdemos - (No table of contents file)
shared\hdlshared - HDL Library
ident\ident - System Identification Toolbox
ident\idobsolete - (No table of contents file)
ident\idguis - (No table of contents file)
ident\idutils - (No table of contents file)
ident\iddemos - (No table of contents file)
ident\idhelp - (No table of contents file)
images\images - Image Processing Toolbox
images\imuitools - Image Processing Toolbox --- imuitools
images\imdemos - Image Processing Toolbox --- demos and sample images
images\iptutils - Image Processing Toolbox --- utilities
imaq\imaq - Image Acquisition Toolbox
imaq\imaqdemos - Image Acquisition Toolbox.
imaqblks\imaqblks - (No table of contents file)
imaqblks\imaqmasks - (No table of contents file)
imaqblks\imaqmex - (No table of contents file)
instrument\instrument - Instrument Control Toolbox

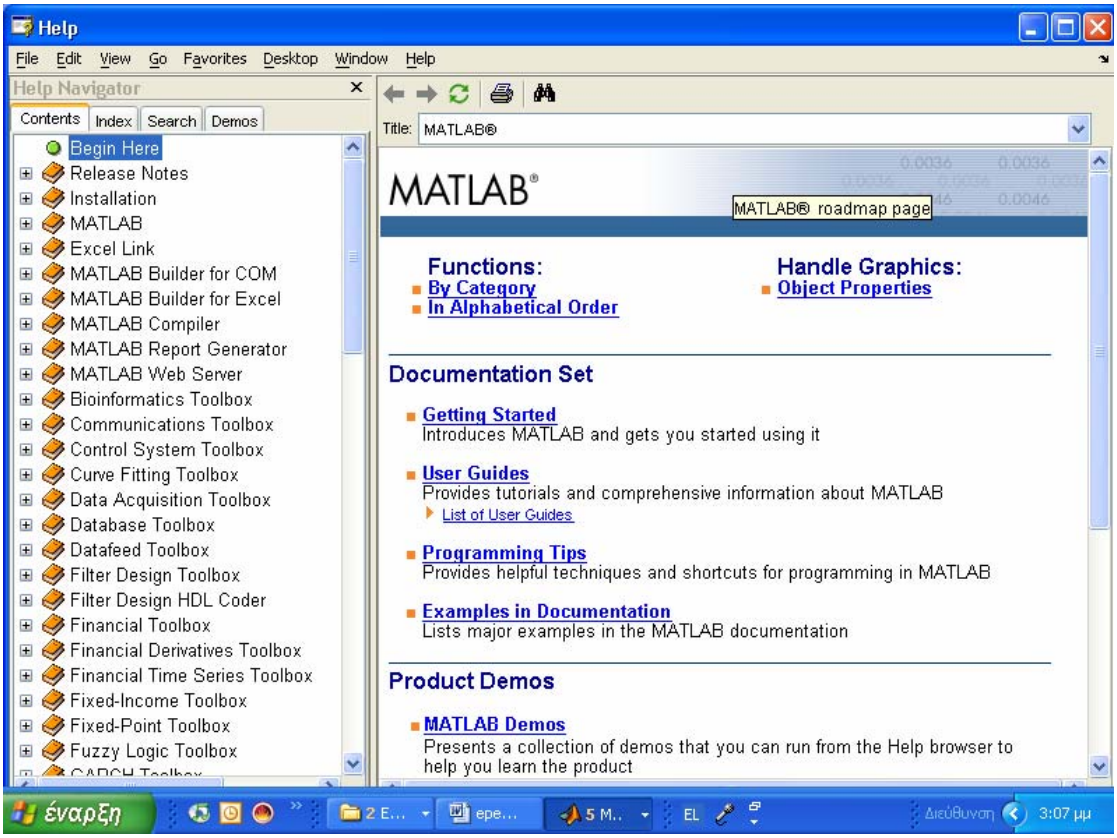
instrument\instrumentdemos - (No table of contents file)
instrumentblks\instrumentblks - (No table of contents file)
instrumentblks\instrumentmex - (No table of contents file)
map\map - Mapping Toolbox
map\mapdemos - Mapping Toolbox Demos and Data Sets.
map\mapdisp - Mapping Toolbox Map Definition and Display.
map\mapformats - Mapping Toolbox File Formats.
map\mapproj - Mapping Toolbox Projections.
mbc\mbc - Model-Based Calibration Toolbox
mbc\mbcdata - Model-Based Calibration Toolbox.
mbc\mbcdesign - Model-Based Calibration Toolbox.
mbc\mbcexpr - Model-Based Calibration Toolbox.
mbc\mbcguitools - Model-Based Calibration Toolbox.
mbc\mbclayouts - (No table of contents file)
mbc\mbcmodels - Model-Based Calibration Toolbox.
mbc\mbcsimulink - Model-Based Calibration Toolbox.
mbc\mbctools - Model-Based Calibration Toolbox.
mbc\mbcview - Model-Based Calibration Toolbox.
mech\mech - SimMechanics
mech\mechdemos - SimMechanics Demos.
pmimport\pmimport - (No table of contents file)
modelsim\modelsim - Link for ModelSim
modelsim\modelsimdemos - (No table of contents file)
mpc\mpc - Model Predictive Control Toolbox
mpc\mpcdemos - (No table of contents file)
mpc\mpcguis - (No table of contents file)
mpc\mpcobsolete - Contents of the previous (obsolete) version of the MPC Toolbox
mpc\mpcutils - (No table of contents file)
shared\slcontrollib - Simulink Control Design Library
targets\mpc555dk - (No table of contents file)
common\configuration - (No table of contents file)
mpc555dk\mpc555demos - (No table of contents file)
mpc555dk\mpc555dk - Embedded Target for Motorola MPC555
mpc555dk\pil - (No table of contents file)
blockset\mfiles - (No table of contents file)
rt\blockset - (No table of contents file)
nnet\nnet - Neural Network Toolbox
nnet\nnutils - (No table of contents file)
nnet\nncontrol - Neural Network Toolbox Control System Functions.
nnet\nndemos - Neural Network Demonstrations.
nnet\nnobsolete - (No table of contents file)
opc\opc - OPC Toolbox
opc\opcgui - (No table of contents file)
opc\opcdemos - (No table of contents file)
toolbox\optim - Optimization Toolbox
osek\osek - Embedded Target for OSEK VDX
osek\osekdemos - (No table of contents file)
osek\blocks - (No table of contents file)
osek\osekworks - (No table of contents file)
osek\proosek - (No table of contents file)
toolbox\pde - Partial Differential Equation Toolbox
powersys\powersys - SimPowerSystems
powersys\powerdemo - SimPowerSystems Demos
drives\drives - (No table of contents file)
drives\drivesdemo - (No table of contents file)
facts\facts - (No table of contents file)
facts\factsdemo - (No table of contents file)
dr\dr - (No table of contents file)
dr\drdemo - (No table of contents file)
simulink\reqmgt -

reqmgt\rmidemos - (No table of contents file)
rf\rf - RF Toolbox
rf\rfdemos - RF Toolbox Demos.
rf\rftool - RF Tool (GUI)
rfblks\rfblks - RF Blockset
rfblks\rfblksmasks - RF Blockset mask helper functions.
rfblks\rfblksmex - RF Blockset S-functions.
rfblks\rfblksdemos - RF Blockset Demos.
robust\robust - Robust Control Toolbox
robust\rctutil - (No table of contents file)
robust\rctdemos - (No table of contents file)
rctobsolete\lmi - Robust Control Toolbox - LMI Solvers.
mutools\commands - (No table of contents file)
mutools\subs - (No table of contents file)
rptgen\rptgen - MATLAB Report Generator
rptgen\rptgendemos - (No table of contents file)
rptgenext\rptgenext - Simulink Report Generator
rptgenext\rptgenextdemos - (No table of contents file)
signal\signal - Signal Processing Toolbox
signal\sigtools - (No table of contents file)
signal\sptoolgui - (No table of contents file)
signal\sigdemos - Signal Processing Toolbox Demonstrations.
slcontrol\slcontrol - Simulink Control Design
slcontrol\slctrlguis - (No table of contents file)
slcontrol\slctrlutil - (No table of contents file)
slcontrol\slctrldemos - (No table of contents file)
slestim\slestdemos - Simulink Parameter Estimation Demos
slestim\slestguis - (No table of contents file)
slestim\slestim - Simulink Parameter Estimation
slestim\slestmex - Simulink Parameter Estimation S-Function MEX-files.
slestim\slestutil - (No table of contents file)
sloptim\sloptim - Simulink Response Optimization
sloptim\sloptguis - (No table of contents file)
sloptim\sloptdemos - Simulink Response Optimization Demos.
sloptim\sloptobsolete - (No table of contents file)
simulink\slvnv - Simulink Verification and Validation
simulink\simcoverage -
simcoverage\simcovdemos - (No table of contents file)
toolbox\splines - Spline Toolbox
toolbox\stats - Statistics Toolbox
toolbox\symbolic - Symbolic Math Toolbox
tic2000\tic2000 - Embedded Target for TI C2000 DSP(tm)
tic2000\tic2000blks - (No table of contents file)
tic2000\tic2000demos - (No table of contents file)
etargets\etargets - (No table of contents file)
tic6000\tic6000 - Embedded Target for TI C6000 DSP(tm)
tic6000\tic6000blks - TI C6000 (tm) Blocks
tic6000\tic6000demos - Embedded Target for TI C6000 DSP(tm) Demos
etargets\rtdxblks - RTDX (tm) Blocks
vr\vr - Virtual Reality Toolbox
vr\vrdemos - Virtual Reality Toolbox examples.
wavelet\wavelet - Wavelet Toolbox
wavelet\wavedemo - Wavelet Toolbox Demonstrations.
webserver\webserver - MATLAB Web Server
webserver\wsdemos - (No table of contents file)
xpc\xpc - xPC Target
build\xpcblocks - (No table of contents file)
xpc\xpcdemos - xPC Target -- demos and sample script files.
xpc\xpcmngr - (No table of contents file)
kernel\embedded - xPC Target Embedded Option

>> Η εντολή *help* εκτελείται και από το μενού της MATLAB επιλέγωντας *help*-> MATLAB *help* (Εικόνα 1.2, Εικόνα 1.3).



Εικόνα 1.2



Εικόνα 1.3

Οι συναρτήσεις που περιέχονται σε κάθε ένα κατάλογο ή εργαλειοθήκη εμφανίζονται δίνοντας την εντολή *help όνομα_καταλόγου*. Για παράδειγμα, αν θέλουμε να δούμε τις συναρτήσεις που περιέχονται στην εργαλειοθήκη control, δίνουμε:

```
>> help control
  Control System Toolbox
  Version 6.1 (R14SP1) 05-Sep-2004

  General.
    ctrlpref    - Set Control System Toolbox preferences.
    ltimodels   - Detailed help on the various types of LTI models.
    ltiprops    - Detailed help on available LTI model properties.

  Creating linear models.
    tf          - Create transfer function models.
    zpk         - Create zero/pole/gain models.
    ss, dss     - Create state-space models.
    frd         - Create a frequency response data models.
    filt        - Specify a digital filter.
    lti/set     - Set/modify properties of LTI models.

  Data extraction.
    tfdata      - Extract numerator(s) and denominator(s).
    zpkdata     - Extract zero/pole/gain data.
    ssdata      - Extract state-space matrices.
    dssdata     - Descriptor version of SSDATA.
    frdata      - Extract frequency response data.
    lti/get     - Access values of LTI model properties.
```

Conversions.
  tf   - Conversion to transfer function.
  zpk   - Conversion to zero/pole/gain.
  ss   - Conversion to state space.
  frd   - Conversion to frequency data.
  chgunits   - Change units of FRD model frequency points.
  c2d   - Continuous to discrete conversion.
  d2c   - Discrete to continuous conversion.
  d2d   - Resample discrete-time model.

System interconnections.
  append   - Group LTI systems by appending inputs and outputs.
  parallel   - Generalized parallel connection (see also overloaded +).
  series   - Generalized series connection (see also overloaded *).
  feedback   - Feedback connection of two systems.
  lft   - Generalized feedback interconnection (Redheffer star product).
  connect   - Derive state-space model from block diagram description.

System gain and dynamics.
  dcgain   - D.C. (low frequency) gain.
  bandwidth   - System bandwidth.
  lti/norm   - Norms of LTI systems.
  pole, eig   - System poles.
  zero   - System (transmission) zeros.
  pzmap   - Pole-zero map.
  iopzmap   - Input/output pole-zero map.
  damp   - Natural frequency and damping of system poles.
  esort   - Sort continuous poles by real part.
  dsort   - Sort discrete poles by magnitude.
  stabsep   - Stable/unstable decomposition.
  modsep   - Region-based modal decomposition.

Time-domain analysis.
  ltiview   - Response analysis GUI (LTI Viewer).
  step   - Step response.
  impulse   - Impulse response.
  initial   - Response of state-space system with given initial state.
  lsim   - Response to arbitrary inputs.
  gensig   - Generate input signal for LSIM.
  covar   - Covariance of response to white noise.

Frequency-domain analysis.
  ltiview   - Response analysis GUI (LTI Viewer).
  bode   - Bode diagrams of the frequency response.
  bodemag   - Bode magnitude diagram only.
  sigma   - Singular value frequency plot.
  nyquist   - Nyquist plot.
  nichols   - Nichols plot.
  margin   - Gain and phase margins.
  allmargin   - All crossover frequencies and related gain/phase margins.
  freqresp   - Frequency response over a frequency grid.
  evalfr   - Evaluate frequency response at given frequency.
  frd/interp - Interpolates frequency response data.

Classical design.
  sisotool   - SISO design GUI (root locus and loop shaping techniques).
  rlocus   - Evans root locus.

Pole placement.

```
  place       - MIMO pole placement.
  acker       - SISO pole placement.
  estim       - Form estimator given estimator gain.
  reg         - Form regulator given state-feedback and estimator gains.

LQR/LQG design.
  lqg         - Single-step LQG design.
  lqr, dlqr   - Linear-quadratic (LQ) state-feedback regulator.
  lqry        - LQ regulator with output weighting.
  lqrd        - Discrete LQ regulator for continuous plant.
  kalman      - Kalman estimator.
  kalmd       - Discrete Kalman estimator for continuous plant.
  lqgreg      - Form LQG regulator given LQ gain and Kalman estimator.
  augstate    - Augment output by appending states.

State-space models.
  rss, drss   - Random stable state-space models.
  ss2ss       - State coordinate transformation.
  canon       - State-space canonical forms.
  ctrb        - Controllability matrix.
  obsv        - Observability matrix.
  gram        - Controllability and observability gramians.
  ssbal       - Diagonal balancing of state-space realizations.
  balreal     - Gramian-based input/output balancing.
  modred      - Model state reduction.
  minreal     - Minimal realization and pole/zero cancellation.
  sminreal    - Structurally minimal realization.

Time delays.
  hasdelay    - True for models with time delays.
  totaldelay  - Total delay between each input/output pair.
  delay2z     - Replace delays by poles at z=0 or FRD phase shift.
  pade        - Pade approximation of time delays.

Model dimensions and characteristics.
  class       - Model type ('tf', 'zpk', 'ss', or 'frd').
  size        - Model sizes and order.
  lti/ndims   - Number of dimensions.
  lti/isempty - True for empty models.
  isct        - True for continuous-time models.
  isdt        - True for discrete-time models.
  isproper    - True for proper models.
  issiso      - True for single-input/single-output models.
  reshape     - Reshape array of linear models.

Overloaded arithmetic operations.
  + and -     - Add and subtract systems (parallel connection).
  *           - Multiply systems (series connection).
  \           - Left divide -- sys1\sys2 means inv(sys1)*sys2.
  /           - Right divide -- sys1/sys2 means sys1*inv(sys2).
  ^           - Powers of a given system.
  '           - Pertransposition.
  .'          - Transposition of input/output map.
  [..]        - Concatenate models along inputs or outputs.
  stack       - Stack models/arrays along some array dimension.
  lti/inv     - Inverse of an LTI system.
  conj        - Complex conjugation of model coefficients.

Matrix equation solvers.
  lyap, dlyap         - Solve Lyapunov equations.
```

```
  lyapchol, dlyapchol - Square-root Lyapunov solvers.
  care, dare        - Solve algebraic Riccati equations.
  gcare, gdare       - Generalized Riccati solvers.
  bdschur           - Block diagonalization of a square matrix.

 Demonstrations.
  Type "demo" or "help ctrldemos" for a list of available demos.


control is both a directory and a function.

 --- help for modeldev/control.m ---

 MODELDEV/CONTROL
```

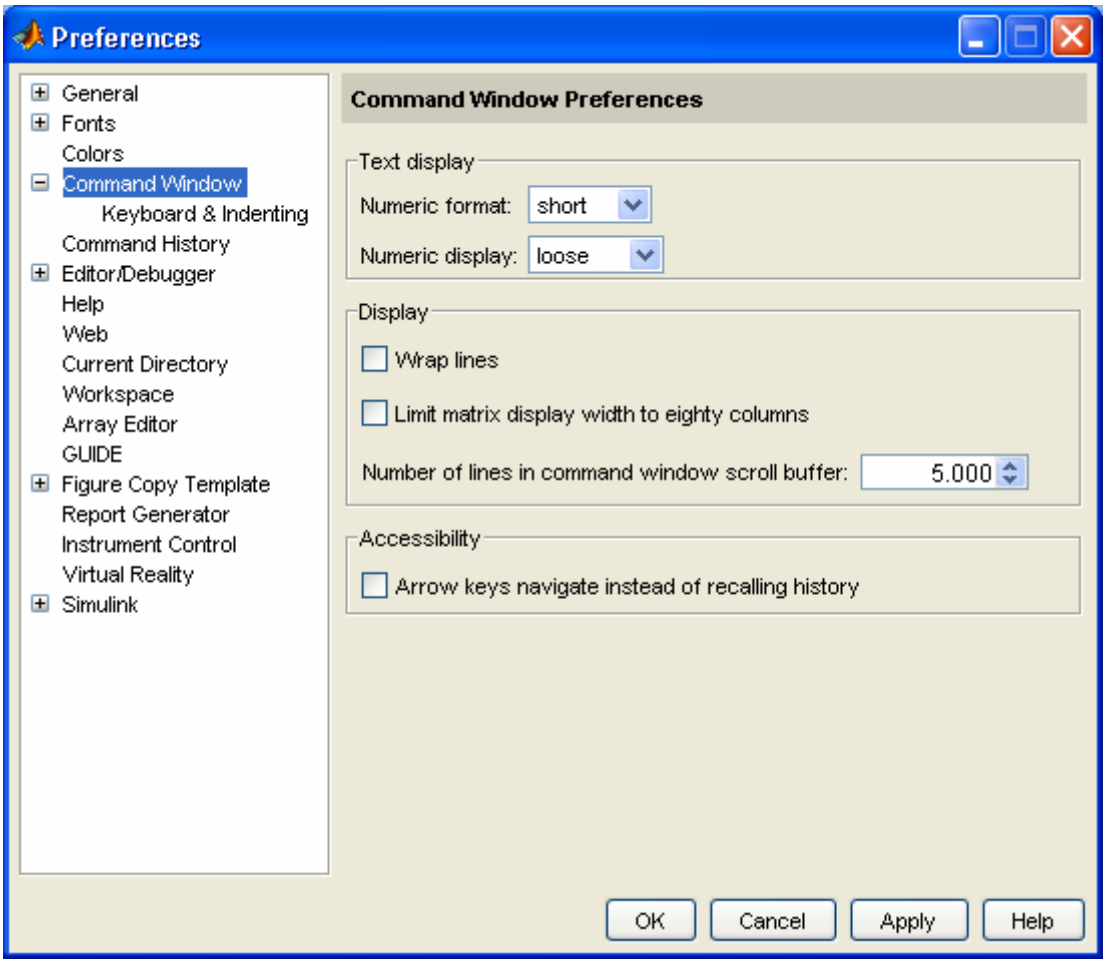
Για να πάρουμε πληροφορίες για μια συγκεκριμένη συνάρτηση, δίνουμε την εντολή *help όνομα_συνάρτησης*. Για παράδειγμα, αν θέλουμε πληροφορίες για τη συνάρτηση tf δίνουμε

```
>> help tf
 TF  Creation of transfer functions or conversion to transfer function.

  Creation:
    SYS = TF(NUM,DEN) creates a continuous-time transfer function SYS with
    numerator(s) NUM and denominator(s) DEN.  The output SYS is a TF object.

    SYS = TF(NUM,DEN,TS) creates a discrete-time transfer function with
    sample time TS (set TS=-1 if the sample time is undetermined).

    S = TF('s') specifies the transfer function H(s) = s (Laplace variable).
    Z = TF('z',TS) specifies H(z) = z with sample time TS.
    You can then specify transfer functions directly as rational expressions
    in S or Z, e.g.,
       s = tf('s');  H = (s+1)/(s^2+3*s+1)

    SYS = TF creates an empty TF object.
    SYS = TF(M) specifies a static gain matrix M.

    In all syntax above, the input list can be followed by pairs
       'PropertyName1', PropertyValue1, ...
    that set the various properties of TF models (type LTIPROPS for details).
    To make SYS inherit all its LTI properties from an existing LTI model
    REFSYS, use the syntax SYS = TF(NUM,DEN,REFSYS).

  Data format:
    For SISO models, NUM and DEN are row vectors listing the numerator and
    denominator coefficients in
     * descending powers of s or z by default
     * ascending powers of q = z^-1 if the 'Variable' property is set to
       'z^-1' or 'q' (DSP convention).

    For MIMO models with NY outputs and NU inputs, NUM and DEN are NY-by-NU
    cell arrays of row vectors where NUM{i,j} and DEN{i,j} specify the
    transfer function from input j to output i.  For example,
       H = TF( {-5 ; [1 -5 6]} , {[1 -1] ; [1 1 0]})
    specifies the two-output, one-input transfer function
```

```
    [     -5 /(s-1)      ]
    [ (s^2-5s+6)/(s^2+s) ]

  By default, transfer functions are displayed as functions of 's' or 'z'.
  Alternatively, you can set the variable name to 'p' (continuous time)
  and 'z^-1' or 'q' (discrete time) by modifying the 'Variable' property.

Arrays of transfer functions:
  You can create arrays of transfer functions by using ND cell arrays for
  NUM and DEN above.  For example, if NUM and DEN are cell arrays of size
  [NY NU 3 4], then
     SYS = TF(NUM,DEN)
  creates the 3-by-4 array of transfer functions
     SYS(:,:,k,m) = TF(NUM(:,:,k,m),DEN(:,:,k,m)),  k=1:3,  m=1:4.
  Each of these transfer functions has NY outputs and NU inputs.

  To pre-allocate an array of zero transfer functions with NY outputs
  and NU inputs, use the syntax
     SYS = TF(ZEROS([NY NU k1 k2...])) .

Conversion:
  SYS = TF(SYS) converts an arbitrary LTI model SYS to the transfer
  function representation. The result is a TF object.

  SYS = TF(SYS,'inv') uses a fast algorithm for conversion from state
  space to TF, but is typically less accurate for high-order systems.

See also ltimodels, filt, set, get, tfdata, subsref, ltiprops, zpk, ss.


 Reference page in Help browser
    doc control/tf
```

Η τελευταία γραμμή της απάντησης παραθέτει τα ονόματα συναρτήσεων που είναι συναφείς με αυτή για την οποία ζητήθηκε βοήθεια.

Μπορούμε να επελέξουμε τη γραμματοσειρά με την οποία θα εμφανίζονται οι εντολές και τα σχόλια στην οθόνη και να προσδιορίσουμε διάφορους παραμέτρους. Με την επιλογή File-Preferences εμφανίζεται το παρακάτω παράθυρο (Εικόνα 1.4). Βλέπουμε ότι υπάρχουν επιλογές για γραμματοσειρές, μέγεθος γραμμάτων, χρώμα φόντου και πολλές άλλες τις οποίες γνωρίζει κάποιος καλύτερα με δοκιμές.



Εικόνα 1.4

1.2. Ορισμοί μεταβλητών και βασικές πράξεις

Κάθε φορά που εκκινείται η MATLAB δημιουργείται στη μνήμη του υπολογιστή ο *χώρος εργασίας* (workspace) εντός του οποίου αποθηκεύονται οι οριζόμενες στο παράθυρο εντολών μεταβλητές. Οι μεταβλητές αυτές είναι διαθέσιμες μέχρι την έξοδο από τη MATLAB, ενώ είναι δυνατή η αποθήκευση τους στο δίσκο και η ανάκτηση τους σε επόμενη εκκίνηση της MATLAB. Ο εξ' ορισμού τύπος μεταβλητής είναι πραγματικός διπλής ακρίβειας. Άλλοι τύποι μεταβλητών που θα χρησιμοποιηθούν στα πλαίσια των εργαστηριακών ασκήσεων είναι ο μιγαδικός αριθμός και η μεταβλητή-αντικείμενο συνάρτησης μεταφοράς. Είναι δυνατή η χρήση

και άλλων τύπων μεταβλητών, η περιγραφή των οποίων είναι εκτός του σκοπού του παρόντος οδηγού.

Η δήλωση μιας μεταβλητής στο παράθυρο εντολών γίνεται ταυτόχρονα με την απόδοση τιμής σε αυτήν. Για παράδειγμα, ο ορισμός μιας μεταβλητής a στην οποία αποδίδουμε την τιμή 10 γίνεται με την εντολή

```
» a=10

a =
   10
```

⇒ *Αν δεν επιθυμούμε να εμφανίζεται το αποτέλεσμα της εκτέλεσης μιας εντολής, βάζουμε στο τέλος της εντολής ένα ελληνικό ερωτηματικό*:

```
» A=5;
```

Για να δούμε ποιές μεταβλητές υπάρχουν στο χώρο εργασίας, δίνουμε την εντολή

```
» who
Your variables are:
A       a
```

⇒ *Η MATLAB κάνει διάκριση πεζών-κεφαλαίων στα ονόματα μεταβλητών.*

Για να δούμε την τιμή μιας μεταβλητής, δίνουμε το όνομα της στη γραμμή εντολής:

```
» A
A =
   5
```

Η διαγραφή μιας μεταβλητής από το χώρο εργασίας γίνεται δίνοντας την εντολή *clear όνομα_μεταβλητής*. Αν θέλουμε να διαγράψουμε όλες τις υπάρχουσες μεταβλητές, δίνουμε απλώς *clear* (προσοχή στη χρήση...)

Μέχρι στιγμής είδαμε τον τρόπο ορισμού μιας βαθμωτής πραγματικής μεταβλητής. Στη συνέχεια παρουσιάζεται ο τρόπος ορισμού πραγματικών διανυσμάτων και πινάκων.

- Ορισμός διανύσματος γραμμής:

```
» a=[1 2 3]

a =
    1    2    3
```

- Ορισμός διανύσματος στήλης:

```
» b=[1;2;3]

b =
    1
    2
    3
```

- Ορισμός πίνακα:

```
» A=[1 2 3;4 5 6]

A =
    1    2    3
    4    5    6
```

- Ορισμός διανύσματος x τα στοιχεία του οποίου ανήκουν στο διάστημα [a,b] και απέχουν μεταξύ τους βήμα d: *x=a:d:b*. Για παράδειγμα

```
» x=0:0.1:1

x =
  Columns 1 through 7

         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000

  Columns 8 through 11

    0.7000    0.8000    0.9000    1.0000
```

(το x ορίζεται ως διάνυσμα γραμμής).

- Πρόσβαση σε μεμονωμένο στοιχείο διανύσματος ή πίνακα: χρήση δεικτών (indexes) γραμμής/στήλης. Για παράδειγμα

```
» A(1,3)

ans =
    3
```

- Πρόσβαση σε μια γραμμή (στήλη) πίνακα: μέσω του τελεστή "*:*". Για παράδειγμα

» A(1,:) δίνει την πρώτη γραμμή του A

» A(:,2) δίνει τη δεύτερη στήλη του A

Σημείωση: οι δείκτες γραμμής/στήλης στη MATLAB παίρνουν μόνο θετικές ακέραιες τιμές (όχι 0 όπως συμβαίνει σε κάποιες γλώσσες προγραμματισμού).

Οι αριθμητικοί τελεστές της MATLAB είναι κατασκευασμένοι ώστε να δέχονται ως ορίσματα όχι μόνο βαθμωτές μεταβλητές, αλλά και μεταβλητές τύπου πραγματικού πίνακα ή διανύσματος. Το αποτέλεσμα της χρήσης ενός τελεστή εξαρτάται από τους τύπους των ορισμάτων του. Οι τελεστές και οι ειδικοί χαρακτήρες της MATLAB παρουσιάζονται εκτελώντας την εντολή *help ops*.

Στη συνέχεια δίνονται ορισμένα παραδείγματα χρήσης των βασικών αριθμητικών τελεστών της MATLAB. Για την εκτέλεση των εντολών που ακολουθούν είναι απαραίτητο να έχουν προηγουμένως ορισθεί οι εξής μεταβλητές:

```
a = 2
b = [0 1 2]
c = [1 2 3]
```

» b+c (b-c) Πρόσθεση (αφαίρεση) διανυσμάτων/πινάκων

```
ans =
   1    3    5      (-1   -1   -1)
```

⇒ *Η μεταβλητή ans χρησιμοποιείται εξ' ορισμού για να αποθηκεύσει το αποτέλεσμα εντολής που δεν εκχωρείται σε μεταβλητή οριζόμενη από το χρήστη*.

» b.*c Πολλαπλασιασμός στοιχείων διανυσμάτων

```
ans =
   0    2    6
```

» d=c' Ανάστροφος διανύσματος/πίνακα (τελεστής ')

```
d =
   1
   2
   3
```

» d*b Πολλαπλασιασμός πινάκων

```
ans =
```

```
    0    1    2
    0    2    4
    0    3    6
```

» a*b (ή a.*b) Πολλαπλασιασμός διανύσματος με αριθμό

```
ans =
    0    2    4
```

» b./c Διαίρεση στοιχείων διανυσμάτων

```
ans =
         0    0.5000    0.6667
```

» b/a (ή b./a) Διαίρεση διανύσματος με αριθμό

```
ans =
         0    0.5000    1.0000
```

» b.^a Ύψωση στοιχείων διανύσματος σε δύναμη

```
ans =
    0    1    4
```

⇒ *Αν και τα δυο ορίσματα είναι βαθμωτά, οι τελεστές .* και ./ είναι ισοδύναμοι με τους τελεστές * και /,αντίστοιχα.*

⇒ *Πολλά μηνύματα λάθους κατά την εκτέλεση πράξεων οφείλονται στην ασυμφωνία διαστάσεων των ορισμάτων.*

Παράδειγμα (προς αποφυγή...): απόπειρα πρόσθεσης διανύσματος γραμμής με διάνυσμα στήλης

```
» b+d
??? Error using ==> +
Matrix dimensions must agree.
```

1.3. Μιγαδικοί Αριθμοί

Αντίθετα με τις άλλες με τις άλλες γλώσσες προγραμματισμού στο MATLAB δεν χρειάζεται να δηλωθεί ο τύπος των μεταβλητών και των αριθμών. Με άλλα λόγια δεν χρειάζεται να δηλώνεται αν μια μεταβλητή θα περιέχει ακέραιους, πραγματικούς ή

μιγαδικούς αριθμούς. Εξ ορισμού στο MATLAB όλες οι μεταβλητές θεωρούνται ότι είναι μιγαδικοί αριθμοί. Έτσι οι πράξεις με πραγματικούς αριθμούς θεωρούνται ειδικές περιπτώσεις των πράξεων μιγαδικών αριθμών. Επίσης, στις παραστάσεις μπορούμε άνετα να αναμιγνύουμε πραγματικούς με μιγαδικούς αριθμούς.
Για την παράσταση μιγαδικών αριθμών χρησιμοποιούνται τα γράμματα i και j τα οποία εξ' ορισμού παριστάνουν την φανταστική μονάδα.

```
>> i % Αυτή είναι η φανταστική μονάδα MATLAB

ans =

      0 + 1.0000i

>> j % Αυτή είναι η φανταστική μονάδα

ans =

      0 + 1.0000i
```

Τα γράμματα i και j μπορούν να θεωρηθούν σαν ιδιόμορφες μεταβλητές. Από τη μια μεριά έχουν όλα τα χαρακτηριστικά των εσωτερικών μεταβλητών του MATLAB, όπως για παράδειγμα οι μεταβλητές pi και eps.

```
>> z1=3+2*i

z1 =

     + 2.0000i
```

Από την άλλη όμως έχουν χαρακτηριστικά που τις διαφοροποιούν σημαντικά. Για παράδειγμα, όταν βρίσκονται στο τέλος ενός αριθμού τον μετατρέπουν σε μιγαδικό

```
z2 =

   3.0000 + 2.0000i
```

```
>> 2j

ans =

      0 + 2.0000i
```

Η ιδιότητα αυτή χάνεται όταν τα i και j βρίσκονται στο τέλος μεταβλητών

```
>> a=2;
>> ai
??? Undefined function or variable 'ai'.
```

Μπορούμε να αλλάξουμε τις τιμές των i και j όπως και όλων των εσωτερικών μεταβλητών. Όταν όμως τις διαγράψουμε από το χώρο εργασίας οι εξ ορισμού τιμές επαναφέρονται. Υπενθυμίζεται ότι αυτή η ιδιότητα δεν ισχύει για τις μεταβλητές που ορίζει ο χρήστης.

```
>> i=3

i =

    3

>> j=2+3j

j =

  2.0000 + 3.0000i

>> clear i j
>> i,j

ans =

      0 + 1.0000i


ans =
```

```
      0 + 1.0000i

>> x=5

x =

    5

>> clear x
>> x
??? Undefined function or variable 'x'.
```

Το MATLAB εκτελεί όλες τις πράξεις των μιγαδικών αριθμών. Στην πραγματικότητα, οι πέντε αριθμητικές πράξεις,

+, -, *, / ή \, ^

ισχύουν για μιγαδικούς αριθμούς. Να μερικά παραδείγματα

```
>> clear
>> x=2-3i; y=2;
>> z1=x^y

z1 =

  -5.0000 -12.0000i

>> z2=y^x

z2 =

  -1.9480 - 3.4936i

>> z3=x^(z1/z2)

z3 =

 -81.6977 -38.6332i
```

Από τη στιγμή που μια μεταβλητή πάρει τιμή, το MATLAB καταγράφει εσωτερικά αν η τιμή της είναι *μιγαδική (complex) ή αριθμητική (numeric)*. Γνωρίζουμε ότι με τη συνάρτηση class(a) μπορούμε να δούμε το είδος της μεταβλητής a. Η συνάρτηση class όμως δεν κάνει διάκριση ανάμεσα στους πραγματικούς και τους μιγαδικούς αριθμούς.

```
>> class(x)

ans =

double

>> class(y)

ans =

double
```

Η διαφοροποίηση ανάμεσα στους πραγματικούς και τους μιγαδικούς αριθμούς φαίνεται στις πληροφορίες που εμφανίζονται με την εντολή whos.

```
>> clear z1 z2 z3, whos
  Name      Size                Bytes  Class

  ans       1x6                    12  char array
  x         1x1                    16  double array (complex)
  y         1x1                     8  double array

Grand total is 8 elements using 36 bytes
```

Βλέπουμε επίσης ότι μια μιγαδική μεταβλητή καταλαμβάνει διπλάσιο χώρο απ' ότι μια πραγματική μεταβλητή.

1.4. Οι εντολές save, load και diary (αποθήκευση/ανάκτηση μεταβλητών)

Όπως αναφέρθηκε στην αρχή της ενότητας, είναι δυνατή η αποθήκευση μιας μεταβλητής (ή και όλων των μεταβλητών) του χώρου εργασίας στο δίσκο του υπολογιστή, αλλά και η ανάκτηση αποθηκευμένων μεταβλητών από το δίσκο στο χώρο εργασίας. Η αποθήκευση μεταβλητής γίνεται με την εντολή *save,* η οποία συντάσσεται ως

save fname X

όπου *fname* είναι το όνομα αρχείου και *X* η αποθηκευόμενη μεταβλητή. Εξ' ορισμού το αρχείο αποθήκευσης έχει την κατάληξη .mat και είναι δυαδικό (binary). Είναι δυνατή η αποθήκευση και σε αρχείο κειμένου (text) αν δοθεί η εντολή ως

save fname X -ascii

Η ανάκτηση μεταβλητών αποθηκευμένων σε αρχεία τύπου .mat ή αρχεία κειμένου γίνεται με την εντολή *load*. Η εντολή αυτή συντάσσεται ως

load fname

εάν το αρχείο με όνομα *fname* είναι τύπου .mat. Οι ανακτώμενες από το αρχείο μεταβλητές είναι διαθέσιμες στο χώρο εργασίας με τα ονόματα που είχαν χρησιμοποιηθεί κατά την αποθήκευση τους. Στην περίπτωση όπου το αρχείο αποθήκευσης είναι τύπου κειμένου (περιέχει γραμμές αριθμητικών τιμών διαχωρισμένων με κενά), χρησιμοποιείται η σύνταξη

load fname.txt

όπου .txt η κατάληξη του αρχείου. Οι τιμές που περιέχει το αρχείο είναι διαθέσιμες στο χώρο εργασίας ως μια μεταβλητή με όνομα το όνομα του αρχείου χωρίς την κατάληξη.

⇒ *Το fname είναι πλήρες όνομα αρχείου (περιλαμβάνει και τη διαδρομή στο δίσκο)*.

Παράδειγμα:

```
» a=[1 2 3];
» save c:\alpha.dat a –ascii Αποθήκευση του διανύσματος a σε αρχείο κειμένου
» load c:\alpha.dat          Ανάκτηση των περιεχομένων του αρχείου alpha.dat στη μεταβλητή
                             του χώρου εργασίας με όνομα alpha
```

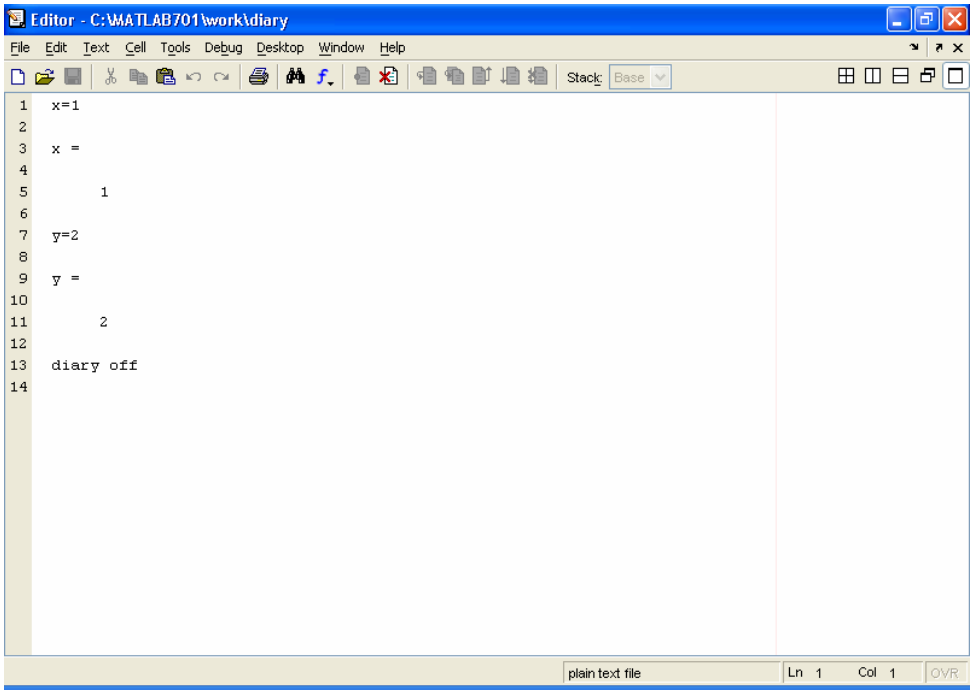
Ένας άλλος τρόπος για αποθήκευση δεδομένων σε δυαδική μορφή (αρχεία τύπου ASCII) είναι με την εντολή *diary*. Από τη στιγμή που θα πληκτρολογηθεί αυτή η εντολή ότι εμφανίζεται στο παράθυρο εντολών αποθηκεύεται σε ένα αρχείο με όνομα *diary*. Η αποθήκευση σταματάει όταν πληκτρολογηθεί *diary off* και ξαναρχίζει όταν πληκτρολογηθεί *diary on*. Τα νέα δεδομένα γράφονται κάτω από τα παλιά. Ας δούμε αυτές τις λειτουργίες στην πράξη.

```
>> clear
>> diary
>> x=1

x =

    1

>> y=2

y =

    2

>> diary off
```

Το αρχείο τοποθετείται στον εξ ορισμού κατάλογο *X:\MATLAB701\work*, όπου X είναι το όνομα του δίσκου στον οποίο είναι εγκαταστημένο το MATLAB. Θα δούμε αργότερα ότι ο κατάλογος αυτός μπορεί να αλλαχτεί από το χρήστη. Μπορούμε να ανοίξουμε το αρχείο με τον κειμενογράφο του MATLAB, τον MATLAB *Editor/Debugger*. Με την εντολή

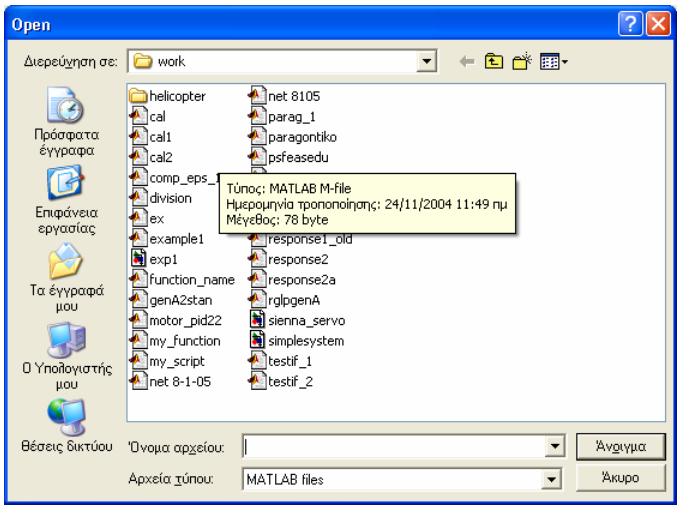
```
open diary
```

εμφανίζεται το παράθυρο του κειμενογράφου με το αρχείο *diary* ανοιχτό (Εικόνα 1.5)



Εικόνα 1.5

Εναλλακτικά μπορούμε να χρησιμοποιήσουμε την επιλογή *File Open*. Τότε εμφανίζεται το παράθυρο ανοίγματος αρχείων (Εικόνα 1.6)



Εικόνα 1.6

Από εκεί μπορούμε στη συνέχεια να ανοίξουμε το αρχείο *diary*.

Με την εντολή *diary on* ανοίγεται το αρχείο *diary* και τα νέα δεδομένα προσαρτώνται στο τέλος των προηγούμενων.

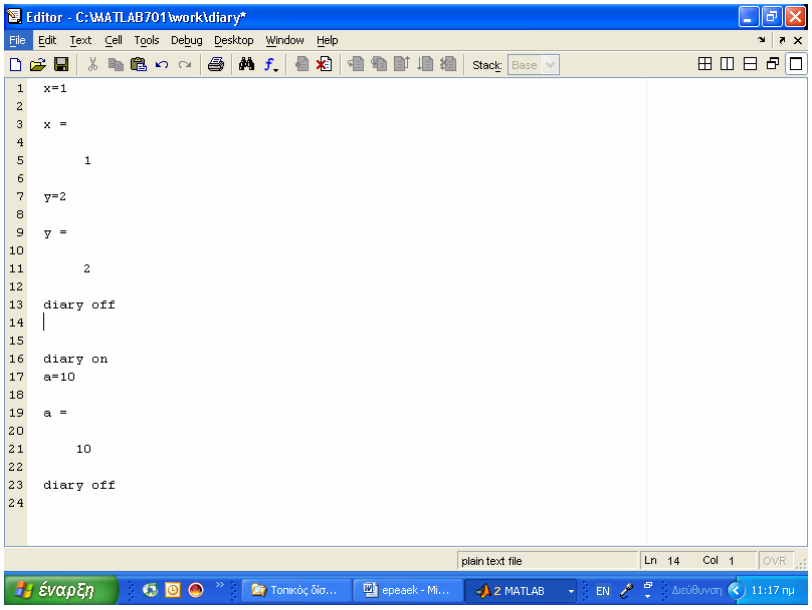
```
>> diary on
>> a=10
```

a =

10

>> diary off

Στη συνέχεια ανοίγοντας το αρχείο *diary* έχουμε (Εικόνα 1.7)



Εικόνα 1.7

Μπορούμε να ορίσουμε εμείς το όνομα του αρχείου γράφοντας το μετά τη λέξη *diary*.

>> diary Session_1_4.dia
>> A=100

A =

100

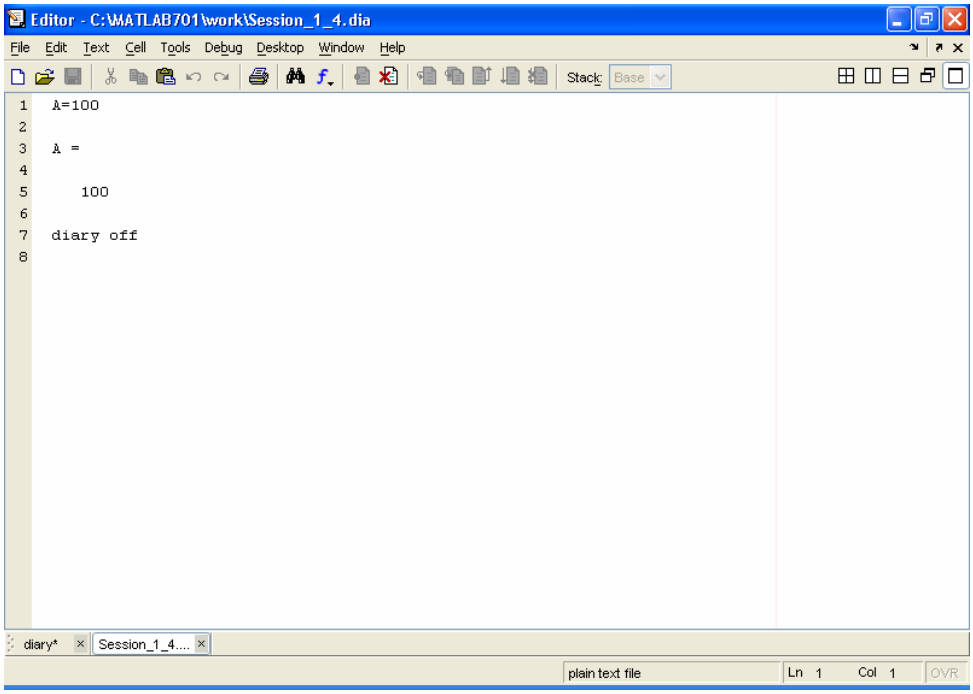
>> diary off
>> B=200

B =

200

Ανοίγοντας το αρχείο *Session_1_4.dia* βλέπουμε (Εικόνα 1.8)

```
>> open  Session_1_4.dia
```



Εικόνα 1.8

Η εντολή *B=200* και η απάντηση δεν αποθηκεύτηκε. Τώρα διαγράφουμε τα αρχεία *diary* και *Session_1_4.dia* με τις εντολές

```
>> delete diary
>> delete Session_1_4.dia
```

1.5. Οι εντολές disp και input

Μπορούμε να εμπλουτίσουμε τα προγράμματα ώστε η λειτουργία τους να είναι κατανοητή και από άλλους που δεν γνωρίζουν τις λεπτομέρειες τους. Για το σκοπό αυτό υπάρχουν οι εντολές *disp* και *input*.

Η εντολή *disp* εμφανίζει στην οθόνη ένα μήνυμα ή την τιμή μιας μεταβλητής χωρίς να εμφανίζεται το σημείο προτροπής. Η εντολή *clc* καθαρίζει την οθόνη.

```
>> clc, clear, x=2; disp(x)
```

2

Με την εντολή *input* μπορούμε να δώσουμε τιμές σε μεταβλητές αφού πρώτα εμφανιστεί στην οθόνη ένα κατάλληλο μήνυμα. Να ένα παράδειγμα.
Πληκτρολογώντας

```
>> r=input('type the radius');
```

Το MATLAB εμφανίζει το μήνυμα

```
type the radius
```

και περιμένει να πληκτρολογηθεί το μήκος της ακτίνας από το πληκτρολόγιο.
Πληκτρολογώντας 100 μπορούμε στη συνέχεια να υπολογίσουμε το εμβαδόν του κύκλου. Θέτοντας

```
>> E=pi*r^2
```

Τότε

```
E =

  3.1416e+004
```

Ή αλλιώς

```
>> E=pi*input('type the radius')^2
type the radius 100

E =

  3.1416e+004
```

ΚΕΦΑΛΑΙΟ 2
ΧΡΗΣΗ ΣΥΝΑΡΤΗΣΕΩΝ MATLAB ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΑΣΚΗΣΕΩΝ.

2.1. Γενικά

Στην ενότητα αυτή δίνεται μια συνοπτική περιγραφή των συναρτήσεων της MATLAB και της εργαλειοθήκης συστημάτων ελέγχου που χρησιμεύουν στην επίλυση ασκήσεων. Περισσότερες πληροφορίες για τις συναρτήσεις αυτές είναι διαθέσιμες μέσω της βοήθειας της MATLAB (κεφάλαιο 1).

Η κλήση μιας συνάρτησης γίνεται από το παράθυρο εντολών της MATLAB με την ακόλουθη γενική σύνταξη:

$$[A_1, A_2, ..., A_n] = function\_name(a_1, a_2, ..., a_m)$$

όπου $[A_1, A_2, ..., A_n]$: η λίστα μεταβλητών στις οποίες επιστρέφονται τα αποτελέσματα της συνάρτησης (ορίσματα εξόδου), *function_name*: το όνομα της συνάρτησης και $a_1, a_2, ..., a_m$: η λίστα ορισμάτων εισόδου της συνάρτησης. Το πλήθος και ο τύπος των ορισμάτων εισόδου και εξόδου εξαρτώνται από την καλούμενη συνάρτηση. Αν η λίστα ορισμάτων εισόδου δεν είναι κατάλληλη, η εντολή κλήσης της συνάρτησης διακόπτεται και παράγεται μήνυμα λάθους. Η λίστα ορισμάτων εξόδου δύναται να είναι κενή, οπότε για την αποθήκευση του επιστρεφόμενου αποτελέσματος χρησιμοποιείται η εξ' ορισμού μεταβλητή *ans*.

Οι περισσότερες από τις συναρτήσεις που περιγράφονται στη συνέχεια επιστρέφουν αποτέλεσμα που εκχωρείται σε μια μεταβλητή, οπότε η εντολή κλήσης τους έχει τη μορφή

$$A = function\_name(a_1, a_2, ..., a_m)$$

ενώ ο τύπος των ορισμάτων και του αποτελέσματος είναι πραγματικό διάνυσμα με αναπαράσταση διπλής ακρίβειας. Στις περιπτώσεις όπου αυτό δεν συμβαίνει, αναφέρονται οι κατάλληλοι τύποι των ορισμάτων ή του αποτελέσματος.

Ας σημειωθεί τέλος ότι οι συναρτήσεις που περιγράφονται στο παρόν δίνονται υπό τη μορφή υπόδειξης: η επίλυση των εργαστηριακών ασκήσεων είναι ενδεχομένως δυνατή και με τη χρήση άλλων συναρτήσεων της MATLAB. Στους πίνακες που ακολουθούν δίνεται η λειτουργία και μια τυπική σύνταξη της εντολής κλήσης για κάθε μια από τις περιγραφόμενες συναρτήσεις.

2.2. Κατασκευή διανυσμάτων

Η εισαγωγή μήτρων με πληκτρολόγηση είναι μια επίπονος διαδικασία ιδιαίτερα στην περίπτωση μεγάλων μήτρων. Για διευκόλυνση των διαδικασιών σε τέτοιες περιπτώσεις το MATLAB διαθέτει ειδικούς τρόπους κατασκευής μήτρων. Μερικοί απ' αυτούς κατασκευάζουν αποκλειστικά διανύσματα ενώ άλλοι κατασκευάζουν γενικές μήτρες δύο διαστάσεων. Για την κατασκευή διανυσμάτων το MATLAB διαθέτει την εντολή *a:b:c* και τις συναρτήσεις *linspace* και *logspace*.

Η εντολή *a:b:c* κατασκευάζει ένα διάνυσμα γραμμή με στοιχεία a, a + b, a + 2b,…., a + kb, όπου k είναι ένας ακέραιος αριθμός. Αν είναι a < c τότε πρέπει να είναι b > 0. Σ' αυτή την περίπτωση το τελευταίο στοιχείο a + kb είναι τέτοιο ώστε a + kb ≤ c, δηλαδή είναι $k = \frac{c-a}{b} = floor\frac{c-a}{b}$. Αν είναι a > c τότε πρέπει να είναι b < 0. Σ' αυτή την περίπτωση το τελευταίο στοιχείο a + kb είναι το μικρότερο στοιχείο τέτοιο ώστε a + kb ≥ c, δηλαδή είναι πάλι $k = \frac{c-a}{c} = floor\frac{c-a}{b}$

```
>> x=1:1:9

x =

   1   2   3   4   5   6   7   8   9

>> y=2:2:9

y =

   2   4   6   8

>> z=5:-3:-9

z =
```

```
    5     2    -1    -4    -7
```

Ο αριθμός b της εντολής *a:b:c* είναι προαιρετικός. Όταν παραλείπεται εννοείται ότι είναι ίσος με 1. Σ' αυτή την περίπτωση είναι αναγκαστικά a < c. Επίσης οι αριθμοί a, b και c μπορούν να αντικατασταθούν με μεταβλητές, συναρτήσεις ή αριθμητικές παραστάσεις.

```
>> x=1:9

x =

    1     2     3     4     5     6     7     8     9

>> x=1:-9

x =

  Empty matrix: 1-by-0

>> w=0:pi:7

w =

         0    3.1416    6.2832

>> p=1:-exp(1):-2*(1+2)

p =

    1.0000   -1.7183   -4.4366
```

Η συνάρτηση *length(x)* υπολογίζει το πλήθος των στοιχείων του διανύσματος X. Από τα προηγούμενα προκύπτει εύκολα ότι είναι length(a:b:c)=k+1. Τέλος αναφέρουμε εδώ ότι το διάνυσμα *a:b:c* συμπεριφέρεται σαν συνάρτηση. Έτσι μπορεί να είναι γραμμή μιας άλλης μήτρας.

```
    >> A=[1:3;4:6;7:9]
```

```
A =

    1    2    3
    4    5    6
    7    8    9

>> x=10:pi:100;
>> y=length(x)

y =

   29

>> k=floor((100-10)/pi)

k =

   28
```

Στην εντολή-συνάρτηση *a:b:c* το πλήθος των στοιχείων δεν είναι γνωστό προκαταβολικά. Η συνάρτηση *lingspace(a,b,c)* κατασκευάζει ένα διάνυσμα γραμμή με πρώτο στοιχείο a, τελευταίο b και ενδιάμεσα άλλα c-2 στοιχεία έτσι ώστε διαδοχικά στοιχεία να ισαπέχουν. Το συνολικό πλήθος των στοιχείων είναι c. Όταν είναι c < 2, η συνάρτηση παράγει τον αριθμό b.

```
>> clear
>> a=linspace(5,8,4)

a =

    5    6    7    8

>> b=linspace(1,2,4)

b =

   1.0000    1.3333    1.6667    2.0000
```

```
>> c=linspace(1,2,-1)

c =

    2
```

Ο αριθμός c είναι προαιρετικός. Όταν δεν αναγράφεται εννοείται ότι είναι c = 100. Τέλος αναφέρουμε εδώ ότι η συνάρτηση κατασκευάζει διάνυσμα ακόμη και όταν a≥ b και ότι μπορεί να είναι γραμμή μιας άλλης μήτρας.

```
>> clear, x=length(linspace(1,2))

x =

   100

>> a=linspace(1,-2,4)

a =

    1     0    -1    -2

>> B=[1:4;a;linspace(1,2,4)]

B =

    1.0000    2.0000    3.0000    4.0000
    1.0000         0   -1.0000   -2.0000
    1.0000    1.3333    1.6667    2.0000

>> c=linspace(1,1,5)

c =

    1     1     1     1     1
```

Προσέξτε τη διαφορά ανάμεσα στο σύμβολο *a:b:c* και στη συνάρτηση *linspace*. Το σύμβολο *a:b:c* είναι εντολή ενώ η *linspace* είναι συνάρτηση αφού έχει μεταβλητές

εισόδου. Όμως τις πιο πολλές φορές η εντολή a:b:c και η συνάρτηση *linspace* μπορούν να χρησιμοποιηθούν εναλλακτικά.

Η συνάρτηση *linspace* κατασκευάζει διανύσματα με στοιχεία που ισαπέχουν. Το MATLAB έχει μια παρόμοια συνάρτηση η οποία κατασκευάζει διανύσματα γραμμές με στοιχεία που ισαπέχουν λογαριθμικά. Είναι η συνάρτηση *logspace(a,b,c)*, η οποία κατασκευάζει ένα διάνυσμα με c στοιχεία, πρώτο στοιχείο 10^a και τελευταίο στοιχείο 10^b. Ακόμη, αν X1, X2 και X3 είναι τρία διαδοχικά στοιχεία του διανύσματος τότε είναι log10(x2)-log10(x1)=log10(x3)-log10(x2). Όπως και στην συνάρτηση *linspace*, το πλήθος, c, των στοιχείων είναι προαιρετικό. Αν δεν αναγράφεται θεωρείται ότι είναι c = 50, όχι 100 που είναι στη *linspace*.

```
>> clear, y=logspace(1,4,4)

y =

        10       100      1000     10000

>> length(logspace(1,100))

ans =

    50
```

2.3. Κατασκευή δισδιάστατων μητρών

Το MATLAB έχει συναρτήσεις για την κατασκευή σχεδόν όλων των γνωστών μητρών από τα μαθηματικά και τις άλλες θετικές επιστήμες. Εδώ θα περιγράψουμε τις πιο βασικές. Ο αναγνώστης μπορεί να χρησιμοποιήσει την υπηρεσία βοήθειας για να διαπιστώσει αν υπάρχουν άλλες συναρτήσεις οι οποίες εξυπηρετούν τους σκοπούς του.

Με το MATLAB μπορούμε να κατασκευάσουμε μήτρες που έχουν όλα τα στοιχεία μηδέν, όλα τα στοιχεία μονάδες και τη μοναδιαία μήτρα. Η συνάρτηση *zeros(m,n)* κατασκευάζει μια μήτρα με m γραμμές και n στήλες και όλα τα στοιχεία μηδέν. Η σύνταξη *zeros(n)* είναι ισοδύναμη με τη σύνταξη *zeros(n,n)*. Παρόμοια είναι η συνάρτηση ones. Η σύνταξη *ones(m,n)* κατασκευάζει μια μήτρα με m γραμμές, n στήλες και όλα τα στοιχεία ίσα με 1. Η σύνταξη *ones(n)* είναι συντόμευση της

εντολής *ones(n,n)*. Η συνάρτηση κατασκευής της μοναδιαίας μήτρας ονομάζεται eye. Η μοναδιαία μήτρα διάστασης n κατασκευάζεται με την εντολή *eye(n)*. Όμως η συνάρτηση *eye* μπορεί να πάρει και δυο μεταβλητές εισόδου. Πληκτρολογώντας *A=eye(m,n)* κατασκευάζεται μια μήτρα με m γραμμές, n στήλες και όλα τα στοιχεία της κυρίας διαγωνίου ίσα με 1, δηλαδή, είναι

$$A(1,1) = A(2,2) = \ldots = 1$$

```
>> zeros(2)

ans =

    0    0
    0    0

>> zeros(2,4)

ans =

    0    0    0    0
    0    0    0    0

>> ones(2)

ans =

    1    1
    1    1

>> ones(2,4)

ans =

    1    1    1    1
    1    1    1    1

>> eye(2)

ans =
```

```
    1    0
    0    1

>> eye(2,4)

ans =

    1    0    0    0
    0    1    0    0
```

Το MATLAB έχει και συναρτήσεις οι οποίες κατασκευάζουν μήτρες τα στοιχεία των οποίων είναι τυχαίοι αριθμοί. Η συνάρτηση *rand(n)* ή *rand(m,n)* κατασκευάζει μήτρες τα στοιχεία των οποίων δημιουργούνται με ομοιόμορφη κατανομή στο διάστημα [0,1], δηλαδή, κάθε αριθμός στο διάστημα [0,1] έχει ίδια πιθανότητα να δημιουργηθεί. Η συνάρτηση *randn(m,n)* ή *randn(n)* κατασκευάζει μήτρες τα στοιχεία των οποίων είναι τυχαίοι αριθμοί που ακολουθούν την κανονική κατανομή N(0,1). Υπενθυμίζεται ότι στην κατανομή N(μ,σ) οι τυχαίοι αριθμοί έχουν μέση τιμή μ και τυπική απόκλιση σ.

```
>> rand(2)

ans =

    0.9501    0.6068
    0.2311    0.4860

>> rand(2,4)

ans =

    0.8913    0.4565    0.8214    0.6154
    0.7621    0.0185    0.4447    0.7919

>> randn(2)

ans =
```

```
   -0.4326    0.1253
   -1.6656    0.2877

>> randn(2,4)

ans =

   -1.1465    1.1892    0.3273   -0.1867
    1.1909   -0.0376    0.1746    0.7258
```

Αναφέρεται εδώ ότι οι τυχαίοι αριθμοί που κατασκευάζονται από τους υπολογιστές δεν είναι πραγματικά τυχαίοι. Κατασκευάζονται από ειδικές συναρτήσεις που ονομάζονται *γεννήτριες τυχαίων αριθμών (random number generators)*. Οι γεννήτριες αυτές κατασκευάζουν ψευδοτυχαίους αριθμούς ξεκινώντας από μια «κατάσταση».Επειδή αυτή η κατάσταση αρχικοποιείται κάθε φορά που ανοίγεται το MATLAB, οι τυχαίοι αριθμοί είναι ίδιοι σε διαφορετικά ανοίγματα του MATLAB. Δοκιμάστε το. Αν θέλουμε να κατασκευάσουμε τους ίδιους τυχαίους αριθμούς δυο και τρεις φορές μπορούμε να το κάνουμε με τη χρήση των εντολών *s=rand('state')* και *rand('state',s)*. Η πρώτη εντολή καταγράφει την τρέχουσα *κατάσταση (state)* της γεννήτριας και η δεύτερη αρχικοποιεί τη γεννήτρια με την κατάσταση s. Π.χ.

Κλείστε και ξανανοίξτε το MATLAB.

```
>> s=rand('state');
>> a=rand(1,5)

a =

    0.9501    0.2311    0.6068    0.4860    0.8913

>> rand('state',s)
>> b=rand(1,5)

b =

    0.9501    0.2311    0.6068    0.4860    0.8913
```

Εκτός από τις προηγούμενες γενικού σκοπού συναρτήσεις κατασκευής μητρών το MATLAB κατασκευάζει και πληθώρα ειδικών μητρών. Οι πιο δημοφιλείς μήτρες έχουν ξεχωριστές συναρτήσεις για την κατασκευή τους. Αυτές είναι οι συναρτήσεις *company, hadamard, hankel, hilb, magic, pascal, rosser, toeplitz, vander* και *wilkinson*. Τις περισσότερες φορές τα ονόματα τους είναι αρκετά για την κατανόηση της λειτουργίας των. Για παράδειγμα η *magic(n)* κατασκευάζει ένα μαγικό τετράγωνο nxn. Όπως είναι γνωστό, ένα μαγικό τετράγωνο nxn έχει στοιχεία τους αριθμούς *1, 2,* …, n^2 έτσι ώστε τα στοιχεία των γραμμών, στηλών και διαγωνίων του να έχουν το ίδιο άθροισμα. Οι μήτρες που κατασκευάζονται με τις προηγούμενες συναρτήσεις είναι τετραγωνικές και οι περισσότερες δέχονται μια μεταβλητή εισόδου, τη διάσταση τους. Μια εξαίρεση αποτελεί η μήτρα *compan* που δέχεται σαν είσοδο ένα διάνυσμα γραμμή. Οι περισσότερες απ' αυτές τις μήτρες έχουν χαρακτηριστικές ιδιότητες. Για παράδειγμα η μήτρα *hilb* (μήτρα του *Hilbert*) έχει στοιχεία (i,j) που δίνονται από τη σχέση $\frac{1}{i+j-1}$ και η αντίστροφη της υπολογίζεται δύσκολα σε πολλές γλώσσες προγραμματισμού. Το MATLAB έχει ειδική συνάρτηση για τον υπολογισμό της αντιστρόφου της μήτρας του *Hilbert*, τη συνάρτηση *invhilb(n)*. Η συνάρτηση για τον υπολογισμό της αντιστρόφου μιας οποιασδήποτε μήτρας ονομάζεται *inv*. Να μερικά παραδείγματα.

```
>> magic(3)

ans =

     8     1     6
     3     5     7
     4     9     2

>> pascal(3)

ans =

     1     1     1
     1     2     3
     1     3     6
```

```
>> hilb(3)

ans =

    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000

>> invhilb(3)

ans =

     9   -36    30
   -36   192  -180
    30  -180   180

>> inv(hilb(3))

ans =

    9.0000  -36.0000   30.0000
  -36.0000  192.0000 -180.0000
   30.0000 -180.0000  180.0000
```

2.4. Ειδικές πράξεις Πινάκων

Στην ενότητα αυτή θα περιγράψουμε πράξεις οι οποίες ορίζονται για πίνακες και έχουν ειδική μορφή. Δύο τέτοιες πράξεις οι οποίες όμως ορίζονται μόνο σε διανύσματα ίσης διάστασης, είναι ο υπολογισμός του εσωτερικού και εξωτερικού γινομένου. Το εσωτερικό γινόμενο δύο διανυσμάτων είναι Χ και Υ είναι ο αριθμός:

$$X_1Y_1 + X_2Y_2 + ... + X_nY_n$$

Και υπολογίζεται από τη συνάρτηση *dot(X,Y)*.Δηλαδή:

```
>> X=1:3, Y=2*ones(1,3)

X =

    1     2     3


Y =

    2     2     2

>> Z=dot(X,Y)

Z =

   12

>>
```

Το εξωτερικό γινόμενο ορίζεται μόνο σε τρισδιάστατα διανύσματα. Δηλαδή το εξωτερικό γινόμενο των διανυσμάτων $X = (X_1, X_2, X_3)$ και $Y = (Y_1, Y_2, Y_3)$ είναι το διάνυσμα $Z = (X_2Y_3 - X_3Y_2, X_3Y_1 - X_1Y_3, X_1Y_2 - X_2Y_1)$

Στο MATLAB η συνάρτηση που υπολογίζει εξωτερικό γινόμενο είναι η *cross(X,Y).*Δηλαδή θα είναι (σύμφωνα με το προηγούμενο παράδειγμα):

```
>> cross(X,Y)

ans =

    -2     4    -2
```

Υπάρχει επίσης μια κατηγορία συναρτήσεων που εκτελούν ειδικές πράξεις με πίνακες. Είναι γνωστό ότι για μια τετραγωνική μήτρα Α μπορούμε να υπολογίσουμε μια άλλη μήτρα Χ ίδιων διαστάσεων έτσι ώστε $X^2 = A$. Θα είναι: $X = \sqrt{A}$

Το MATLAB έχει τη συνάρτηση *sqrt(A)* η οποία όμως υπολογίζει τις τετραγωνικές ρίζες των στοιχείων της μήτρας Α. Η μήτρα Χ υπολογίζεται με τη συνάρτηση *sqrtm(A)*. Δηλαδή:

```
>> A=pascal(3)

A =

     1     1     1
     1     2     3
     1     3     6

>> B=sqrt(A)
```

```
B =

    1.0000    1.0000    1.0000
    1.0000    1.4142    1.7321
    1.0000    1.7321    2.4495

>> X=sqrtm(A)

X =

    0.8775    0.4387    0.1937
    0.4387    1.0099    0.8874
    0.1937    0.8874    2.2749
```

2.5. Πράξεις πολυωνύμων

Τα πολυώνυμα στο MATLAB παριστάνονται με τα διανύσματα των συντελεστών τους. Ειδικότερα το πολυώνυμο βαθμού n

$$a_n X^n + a_{n-1} X^{n-1} + a_1 X + a_0$$

Παριστάνεται με το διάνυσμα

$$[a_n \;\; a_{n-1} \;\; \ldots \; a_1 \;\; a_0]$$

Για παράδειγμα το πολυώνυμο

$$7X^5 + 3X^3 + 4$$

Παριστάνεται με το διάνυσμα

```
[7 0 3 0 0 4];
```

Προσέξτε ότι γράφονται και οι συντελεστές που είναι ίσοι με μηδέν.

2.6. Συνθέσεις Πινάκων

Το MATLAB παρέχει τη δυνατότητα να διαγράψουμε γραμμές ή στήλες σε πίνακες. Στις λειτουργίες αυτές σημαντικό ρόλο παίζει ο κενός πίνακας που συμβολίζεται με *[ ]*.

```
>> clear, a=magic(4)

a =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

>> a(2,:)=[ ] %διαγραφή της γραμμής 2

a =

    16     2     3    13
     9     7     6    12
     4    14    15     1

>> a(:,[1,3])=[ ] %διαγραφή στηλών 1 και 3

a =

     2    13
     7    12
    14     1

>> a(1:2,:)=[ ] %διαγραφή γραμμών 1 μέχρι 2

a =

    14     1
```

Η αντικατάσταση ενός στοιχείου, μιας γραμμής, μιας στήλης ή ενός υποπίνακα γίνεται με παρόμοιο τρόπο.

```
>> clear, A=magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> A(1,3)=1 %αντικατάσταση στοιχείου A(1,,3)

A =
```

```
    8     1     1
    3     5     7
    4     9     2

>> A(1,:)=[1 2 3] %αντικατάσταση πρώτης γραμμής

A =

    1     2     3
    3     5     7
    4     9     2

>> A([1,3],:)=[5 -2 4; 7 -10 3] %αντικατάσταση γραμμών 1 και 3

A =

    5    -2     4
    3     5     7
    7   -10     3
```

Μπορούμε να εισάγουμε γραμμές ή στήλες στο τέλος ενός πίνακα.

```
>> clear, a=[1 2;3 4]

a =

    1     2
    3     4

>> a(:,3)=[5 6]'

a =

    1     2     5
    3     4     6

>> a(:,6)=[7 8]'

a =

    1     2     5     0     0     7
    3     4     6     0     0     8

>> a(:,[4 5])=[ ]

a =

    1     2     5     7
    3     4     6     8

>> a(3,:)=[1 2 3 4]

a =

    1     2     5     7
    3     4     6     8
    1     2     3     4
```

Η εντολή *A(:)*, όπου Α είναι ένας πίνακας κατασκευάζει ένα διάνυσμα στήλη που αποτελείται από όλα τα στοιχεία του πίνακα Α. Στο διάνυσμα γράφονται πρώτα τα στοιχεία της πρώτης στήλης μετά τα στοιχεία της δεύτερης στήλης κ.ο.κ

```
>> A=magic(2)

A =

    1    3
    4    2

>> X=A(:)

X =

    1
    4
    3
    2
```

Υπάρχει η δυνατότητα να παραθέσουμε τον ένα πίνακα δίπλα στον άλλο ή τον ένα κάτω από τον άλλο ώστε να κατασκευάζονται νέοι πίνακες. Σ' αυτή την περίπτωση οι πίνακες που παρατίθενται πρέπει να έχουν συμβατές διαστάσεις.

```
>> A=[1:3], B=[4 5]

A =

    1    2    3


B =

    4    5

>> [A B]

ans =

    1    2    3    4    5

>> C=[1 2 3; 4 5 6], D=magic(2)

C =

    1    2    3
    4    5    6


D =

    1    3
    4    2

>> E=[A B; C D]
```

```
E =

    1    2    3    4    5
    1    2    3    1    3
    4    5    6    4    2
```

Αυτή η δυνατότητα συνδυαζόμενη με την άλλη δυνατότητα του MATLAB, η οποία επιτρέπει ένας πίνακας να βρίσκεται στο αριστερό και στο δεξιό μέρος μιας εντολής καταχώρησης, επιτρέπει την εισαγωγή στηλών ή γραμμών και ολόκληρων πινάκων στο ενδιάμεσο ενός άλλου πίνακα. Για παράδειγμα οι εντολές με τις οποίες παρεμβάλλουμε τα στοιχεία 10, 11 μετά το στοιχείο 3 του διανύσματος *a=[1 3 5 7]* είναι οι εξής:

```
>> a=[ 1 3 5 7];
>> a=[a([1 2]),[10 11], a([3 4])]

a =

    1    3   10   11    5    7
```

Μπορούμε να δημιουργήσουμε ένα κενό πίνακα με την εντολή *a=[ ]* και στη συνέχεια να προσθέσουμε οποιοδήποτε αριθμό γραμμών και / ή στηλών.

```
>> a=[], a=[a [1 2]]

a =

    []


a =

    1    2
```

Όταν ξεκινούμε έναν κενό πίνακα και στη συνέχεια εισάγουμε γραμμές και / ή στήλες το MATLAB υπολογίζει κάθε φορά τη διάσταση του νέου πίνακα που κατασκευάζεται. Αυτό είναι πολύ χρονοβόρο από υπολογιστικής απόψεως. Αν γνωρίζουμε από την αρχή το μέγιστο μέγεθος του πίνακα είναι προτιμότερο να αρχικοποιήσουμε τον πίνακα με τον αρχικό πίνακα.

2.7. Μερικές δημοφιλείς συναρτήσεις

Οι συναρτήσεις του υπολογισμού του μεγίστου ή ελαχίστου μιας πεπερασμένης ακολουθίας αριθμών και η διάταξη των στοιχείων της ακολουθίας σε αύξουσα σειρά είναι από τις πιο δημοφιλείς συναρτήσεις του MATLAB. Το μέγιστο στοιχείο ενός διανύσματος X υπολογίζεται με τη συνάρτηση *max(X),* το ελάχιστο με τη συνάρτηση *min(X)* και η συνάρτηση *sort(X)* ταξινομεί τα στοιχεία του X κατά αύξουσα σειρά.

```
>> X=round(10*rand(1,8)) %εσείς μπορεί να υπολογίσετε άλλο X
X =
   10    2    6    5    9    8    5    0
>> max(X)
ans =
   10
>> min(X)
ans =
    0
>> sort(X)
ans =
    0    2    5    5    6    8    9   10
```

Εκτός από το μέγιστο και το ελάχιστο στοιχείο οι συναρτήσεις min και max υπολογίζουν και τους αντίστοιχους δείκτες. Τότε υπάρχουν δυο μεταβλητές εξόδου. Επίσης το διάνυσμα εισόδου X μπορεί να είναι διάνυσμα στήλη. Παρόμοια είναι η λειτουργία της συνάρτησης sort όταν υπάρχουν δύο μεταβλητές εξόδου. Τα παρακάτω παραδείγματα θα βοηθήσουν στην παραπέρα κατανόηση των λειτουργιών αυτών.

```
>> X=X';
>> [megisto,i]=max(X)
megisto =
   10
```

```
i =

     1

>> [elaxisto,j]=min(X)

elaxisto =

     0


j =

     8

>> [newX,k]=sort(X); %Τα διανύσματα newX και k είναι στήλες
>> newX'

ans =

     0
     2
     5
     5
     6
     8
     9
    10

>> k'

ans =

     8
     2
     4
     7
     3
     6
     5
     1
```

Όταν η μεταβλητή εισόδου είναι πίνακας δύο διαστάσεων, η συνάρτηση *min* υπολογίζει τα ελάχιστα στοιχεία κάθε στήλης. Τα ελάχιστα στοιχεία καταγράφονται σε ένα διάνυσμα γραμμή. Όταν υπάρχουν δυο μεταβλητές εξόδου, στην πρώτη καταγράφει τα ελάχιστα στοιχεία κάθε στήλης και η δεύτερη τις θέσεις τους στις αντίστοιχες στήλες. Ανάλογες λειτουργίες έχει και η συνάρτηση *max*. Με παρόμοιους τρόπους συντάσσεται και η συνάρτηση *sort*. Τα επόμενα παραδείγματα διευκρινίζουν περισσότερο τις λειτουργίες αυτές.

```
>> A=magic(3)

A =
```

```
    8    1    6
    3    5    7
    4    9    2

>> [MEG,IA]=max(A)

MEG =

    8    9    7


IA =

    1    3    2

>> [EL,IJ]=min(A)

EL =

    3    1    2


IJ =

    2    1    3

>> [NEWA,K]=sort(A)

NEWA =

    3    1    2
    4    5    6
    8    9    7


K =

    2    1    3
    3    2    1
    1    3    2
```

Οι συναρτήσεις *min, max* και *sort* μπορούν να δεχτούν δύο μεταβλητές εισόδου. Τότε όμως οι λειτουργίες τους διαφοροποιούνται. Η συνάρτηση *sort(A,dim)*, όπου dim είναι ακέραιος θετικός αριθμός, ενεργεί στη διάσταση *dim*. Έτσι *sort(A,1)* είναι ισοδύναμη με τη *sort(A)* ενώ η συνάρτηση *sort(A,2)* ταξινομεί τις γραμμές του πίνακα A.

```
>> [B,L]=sort(A,2) %δέχεταιμιά ή δύο μεταβλήτες εξόδου

B =

    1    6    8
    3    5    7
    2    4    9
```

```
L =

    2    3    1
    1    2    3
    3    1    2
```

Όταν οι συναρτήσεις *min* και *max* δέχονται δύο μεταβλητές εισόδου, λειτουργούν διαφορετικά. Ας πάρουμε τη συνάρτηση *C=min(A,B),* όπου Α και Β είναι πίνακες ίδιων διαστάσεων. Ο πίνακας C έχει στοιχεία $c_{i,j}$ τέτοια ώστε $c_{ij} = \min\{a_{ij}, b_{ij}\}$. Παρόμοια είναι η λειτουργία της συνάρτησης *max*. Αναφέρουμε επίσης ότι μπορεί μια από τις μεταβλητές να είναι ένας αριθμός και ότι υπάρχει μόνο μια μεταβλητή εξόδου.

```
>> C=min(A,4)

C =

    4    1    4
    3    4    4
    4    4    2

>> D=min(C,[1 2 3; 4 5 6; 7 8 9])

D =

    1    1    3
    3    4    4
    4    4    2
```

Για να υπολογιστούν τα ελάχιστα ή μέγιστα στοιχεία των γραμμών ενός πίνακα Α πρέπει να εισάγουμε τρεις μεταβλητές εισοδου στις συναρτήσεις *min* και *max*. Ο πρώτος είναι ο πίνακας Α, ο δεύτερος ο κενός πίνακας *[ ]* και ο τρίτος ένας ακέραιος αριθμός *dim* που δηλώνει τη διάσταση ως προς την οποία λειτουργεί η συνάρτηση. Έτσι *min(A, [ ], 1)* είναι ίδια με την *min(A)*, ενώ *min(A, [ ], 2)* υπολογίζει τα ελάχιστα στοιχεία των γραμμών της μήτρας Α.

```
>> [E,F]=min(A,[ ],2);E',F'

ans =

    1    3    2


ans =

    2    1    3
```

2.8. Οι συναρτήσεις sum prod και cumsum, cumprod.

Όταν X είναι διάνυσμα η συνάρτηση *sum(X)* υπολογίζει το άθροισμα των στοιχείων του ενώ η συνάρτηση *prod(X)* υπολογίζει το γινόμενο. Άρα θα είναι:

```
>> X=1:10

X =

    1    2    3    4    5    6    7    8    9   10

>> sum(X)

ans =

   55

>> sum_1_to_10=(1+10)*10/2

sum_1_to_10 =

   55

>> Y=2.^[0:3]

Y =

    1    2    4    8

>> prod(Y)

ans =

   64

>> prod_of_powers_of_2=2^(0+1+2+3)

prod_of_powers_of_2 =

   64
```

Οι συναρτήσεις *sum* και *prod* δέχονται σαν μεταβλητή εισόδου οποιοδήποτε πίνακα. Τότε υπολογίζουν τα αθροίσματα ή τα γινόμενα των στοιχείων κάθε στήλης. Τα αποτελέσματα τα γράφουν σε ένα διάνυσμα γραμμή.

```
>> A=pascal(3)

A =

    1    1    1
    1    2    3
    1    3    6
```

```
>> sum(A)

ans =

    3     6    10

>> prod(A)

ans =

    1     6    18
```

Οι συντάξεις *sum(A,1)* και *prod(A,1)* παράγουν τα ίδια αποτελέσματα. Όμως όταν η δεύτερη μεταβλητή εισόδου πάρει την τιμή 2 τότε υπολογίζονται τα αθροίσματα και τα γινόμενα των γραμμών.

```
>> sum(A,2)

ans =

     3
     6
    10

>> prod(A,2)

ans =

     1
     6
    18
```

Παρόμοιες αλλά λίγο πιο γενικές είναι οι συναρτήσεις *cumsum* και *cumprod*. Η συνάρτηση *a=cumsum(X),* όπου X είναι διάνυσμα υπολογίζει ένα διάνυσμα a έτσι ώστε το στοιχείο a(i) είναι άθροισμα των πρώτων i στοιχείων του διανύσματος X.

```
>> cumsum(1:5)

ans =

     1     3     6    10    15

>> cumprod(1:5)

ans =

     1     2     6    24   120

>> A=[1 2 3];
>> cumsum(A)

ans =

     1     3     6
```

```
>> cumprod(A)

ans =

    1     2     6

>> cumsum(A,2)

ans =

    1     3     6

>> cumprod(A,2)

ans =

    1     2     6
```

2.9. Ενδεικτικές συναρτήσεις

C = conv(a,b)	C = το γινόμενο των πολυωνύμων a και b *σημείωση*: *ένα πολυώνυμο ορίζεται στη MATLAB μέσω του διανύσματος με στοιχεία τους συντελεστές του σε φθίνουσα διάταξη, π.χ. το* $a(s) = 2s^2+3s+1$ *ορίζεται ως το διάνυσμα* $a = [2\ 3\ 1]$

Παράδειγμα:

```
>> a=[1 2 3];
>> b=[2 6 7];
>> c=conv(a,b)

c =

    2    10    25    32    21
```

B = roots(a)	B = οι ρίζες του πολυωνύμου a (*εν γένει μιγαδικές*) *σημείωση*: *η δήλωση/αναπαράσταση μιγαδικού αριθμού στη MATLAB γίνεται ως* $x = m+ni$, *όπου* m,n *πραγματικοί*

Παράδειγμα:

```
>> a=[1 4 5];
>> b=roots(a)

b =

  -2.0000 + 1.0000i
  -2.0000 - 1.0000i
```

plot(y)	Σχεδιάζει τα στοιχεία του διανύσματος y συναρτήσει του δείκτη τους
plot(x,y)	Σχεδιάζει το διάνυσμα y συναρτήσει του διανύσματος x
plot(x,y,'r')	Σχεδιάζει το διάνυσμα y συναρτήσει του διανύσματος x και του δίνει ένα συγκεκριμένο χρώμα π.χ.(κόκκινο) (Το 'r' είναι το πρώτο γράμμα της λέξης red:κόκκινο) Παράδειγμα για την συνάρτηση αυτή δίδεται μαζί με τις συναρτήσεις hold on και hold off.

Το MATLAB παρέχει τη δυνατότητα στο χρήστη να χρωματίζει τις γραμμές μιας γραφικής παράστασης με χρώματα της αρεσκείας του.

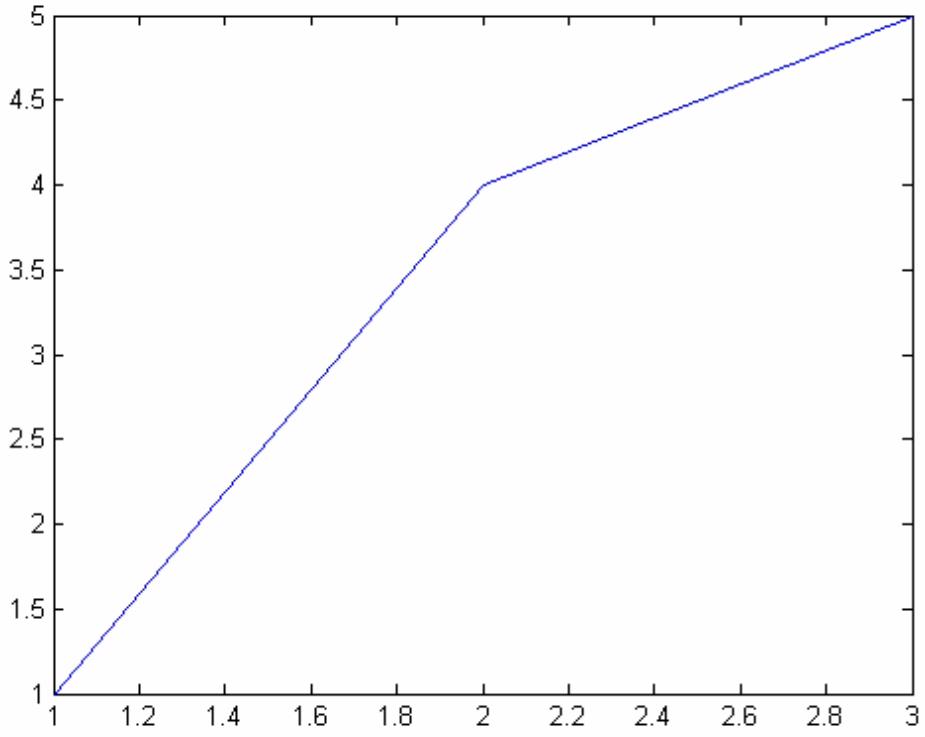
Στη συνέχεια δίνεται ο Πίνακας 2.1 ο οποίος απεικονίζει τα σύμβολα και τις ερμηνείες τους που χρησιμοποιούνται στην εντολή *plot.*

ΧΡΩΜΑ		ΣΗΜΑΔΙΑ		ΜΟΡΦΗ ΓΡΑΜΜΗΣ	
Σύμβολο	**Ερμηνεία**	**Σύμβολο**	**Ερμηνεία**	**Σύμβολο**	**Ερμηνεία**
Y	Κίτρινο	.	Τελεία	-	Συνεχείς
M	Ιώδες	o	Κύκλος	:	Τελείες
C	Κυανό	x	Σημείο x	-.	Τελείες-Παύλες
R	Κόκκινο	+	Σημείο +	--	Διακεκομμένη
G	Πράσινο	*	Αστερίσκος		
B	Γαλάζιο	s	Τετράγωνο		
W	Άσπρο	d	Ρόμβος		
K	Μαύρο	v	Βέλος κάτω		
		^	Βέλος πάνω		
		<	Βέλος αριστερά		
		>	Βέλος δεξιά		
		p	Πεντάγωνο		
		h	Εξάγωνο		

Πίνακας 2.1

Παράδειγμα:

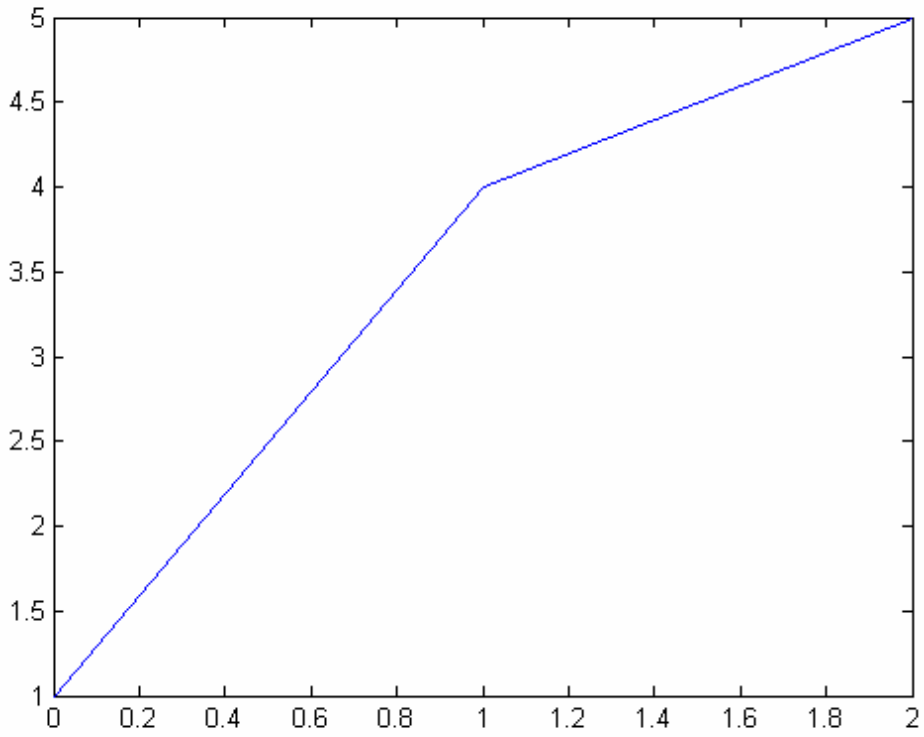
```
>> plot(a)

>>
```



Γράφημα 2.1

```
> t=0:1:2;
>> plot(t,a)
```

>>



Γράφημα 2.2

max(x)	Το μέγιστο στοιχείο του διανύσματος x

Παράδειγμα:

```
>> a=[1 2 3];
>> max(a)

ans =

    3

>>
```

length(x)	Το μήκος του διανύσματος x

Παράδειγμα:

```
>> length(a)

ans =
```

3

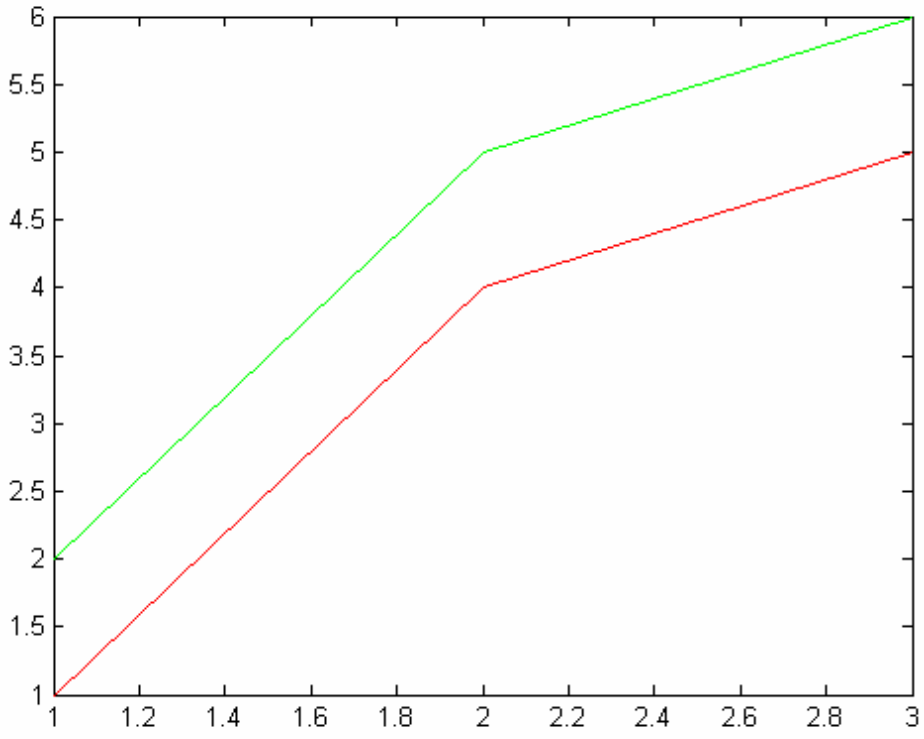
>>

hold on hold off	Η επόμενη εντολή plot θα σχεδιάσει στο ήδη υπάρχον γράφημα Αναίρεση της εντολής hold on (κάθε εντολή plot δημιουργεί νέο γράφημα) Οι συναρτήσεις hold on και hold on μπορούν να συνδυαστούν με τη συνάρτηση plot(x,y,'r') που αναφέρθηκε προηγουμένως.

Παράδειγμα:

```
>> plot(t,y,'r')
>> hold off
>> clear
>> a=[1 4 5];
>> plot(a,'r')
>> hold on
>> b=[2 5 6];
>> plot(b,'g')
>> hold off
>>
```

και τα αντίστοιχα διαγράμματα παρουσιάζονται στο παρακάτω διάγραμμα :



Γράφημα 2.3

rand(m,n)	Ένας πίνακας m×n του οποίου τα στοιχεία είναι τυχαίοι αριθμοί που ακολουθούν ομοιόμορφη κατανομή στο διάστημα (0,1)

Παράδειγμα:

```
>> rand(2,3)

ans =

   0.9501    0.6068    0.8913
   0.2311    0.4860    0.7621

>>
```

abs(s)	Η απόλυτη τιμή των στοιχείων του διανύσματος s. Αν το s είναι μιγαδικό διάνυσμα, το μέτρο των στοιχείων του s.

Παράδειγμα:

```
>> a=[-1 5 -9];
>> abs(a)

ans =

    1     5     9

>>
```

angle(s)	Η φάση των στοιχείων του μιγαδικού διανύσματος s.

Παράδειγμα:

```
>> b=[-1 5 -8];
>> angle(b)

ans =

   3.1416         0    3.1416

>>
```

X = A\b (τελεστής \)	Η λύση του γραμμικού συστήματος AX=b

Παράδειγμα:

```
>> a=[5 6;7 8];
```

```
>> b=[7 4;9 6];
>> X=a\b

X =

   -1    2
    2   -1

>>
```

Εκτός από τις παραπάνω συναρτήσεις υπάρχουν και συναρτήσεις που συναντάμε και σε άλλες γλώσσες προγραμματισμού όπως οι συναρτήσεις *ημίτονο (sin), συνημίτονο (cos), εφαπτομένη (tan), (sec), (csc)* και *συνεφαπτομένη (cot)*. Επίσης που εκτελούν μαθηματικές πράξεις όπως οι *λογαριθμικές* και η συνάρτηση *sqrt* που υπολογίζει την τετραγωνική ρίζα. Η χρήση τους φαίνεται στα παρακάτω παραδείγματα.

```
>> sin(pi/2), cos(pi/2),tan(pi/2),tan(pi/2)

ans =

     1


ans =

  6.1232e-017


ans =

  1.6331e+016


ans =

  1.6331e+016

>> sqrt(25), log(exp(4)),log10(10^4)

ans =
```

```
     5


ans =

     4


ans =

     4

>> pow2(10), log2(pow2(10)), nextpow2(25)

ans =

        1024


ans =

    10


ans =

     5
```

2.10. Συναρτήσεις μιγαδικών αριθμών

Ενθυμούμενοι ότι υπάρχουν δύο παραστάσεις των μιγαδικών αριθμών, η *καρτεσιανή* $a + bi$ και η *πολική* $p = Me^{i\theta}$, όπου a, b, M και θ συνδέονται με τις σχέσεις

$$M = \sqrt{a^2 + b^2}$$
$$\theta = \tau o \xi \varepsilon \phi \left(\frac{b}{a} \right)$$
$$a = M \sigma \upsilon \nu \theta$$
$$b = M \eta \mu \theta$$

Η συνάρτηση *abs(p)* υπολογίζει το μέτρο του μιγαδικού αριθμού *p*. Αν ο *p* είναι πραγματικός αριθμός υπολογίζει την απόλυτη τιμή. Η γωνία θ υπολογίζεται με τη συνάρτηση *angle(p)*.

```
>> a=sqrt(2)/2 % a=cos(-45 μοίρες)

a =

   0.7071

>> b=-sqrt(2)/2 % b=sin(-45 μοίρες)

b =

  -0.7071

>> p=a+b*i % M=1, θ=-45 μοίρες=pi/4

p =

  0.7071 - 0.7071i
```

Υπενθυμίζεται ότι η γωνία θ είναι η γωνία που σχηματίζει ο αριθμός *p* με τον πραγματικό άξονα

```
>> M=abs(p)

M =

    1

>> angle(p)
```

```
ans =

   -0.7854

>> -pi/4

ans =

   -0.7854
```

Βλέπουμε ότι και η συνάρτηση *angle* υπολογίζει τη γωνία σε ακτίνια.
Υπάρχουν και οι πιο απλές συναρτήσεις με τις οποίες υπολογίζεται το πραγματικό μέρος, *real(p),* το φανταστικό μέρος, *imag(p),* και ο συζυγής μιγαδικός αριθμός, *conj(p).*

```
>> x=2-3i;
>> real(x)

ans =

     2

>> imag(x)

ans =

    -3

>> conj(x)

ans =

   2.0000 + 3.0000i
```

Αφήνουμε σαν άσκηση την χρήση των συναρτήσεων *unwrap, isreal, cplxpair*. Η εντολή *help* θα είναι μια πραγματική βοήθεια.

2.11. Συναρτήσεις σχετικές με συστήματα Αυτομάτου Ελέγχου

Sys = tf(num,den)	Sys = *μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς* με αριθμητή και παρανομαστή τα πολυώνυμα num και den, αντίστοιχα

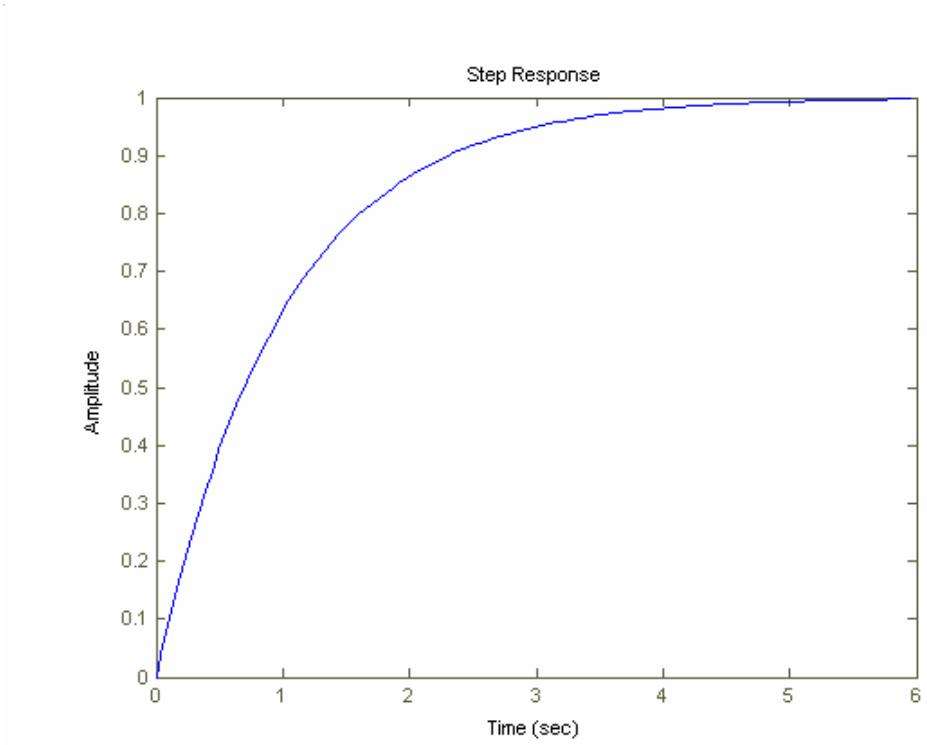
Παράδειγμα:

```
>> num=2;
>> den=[2 2];
>> sys=tf(num,den)

Transfer function:
  2
-------
2 s + 2

>>
```

step(sys)	Υπολογίζεται και σχεδιάζεται η μοναδιαία βηματική απόκριση του συστήματος με συνάρτηση μεταφοράς sys (sys=*μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς)*

Παράδειγμα:

```
>> step(sys)
>>
```

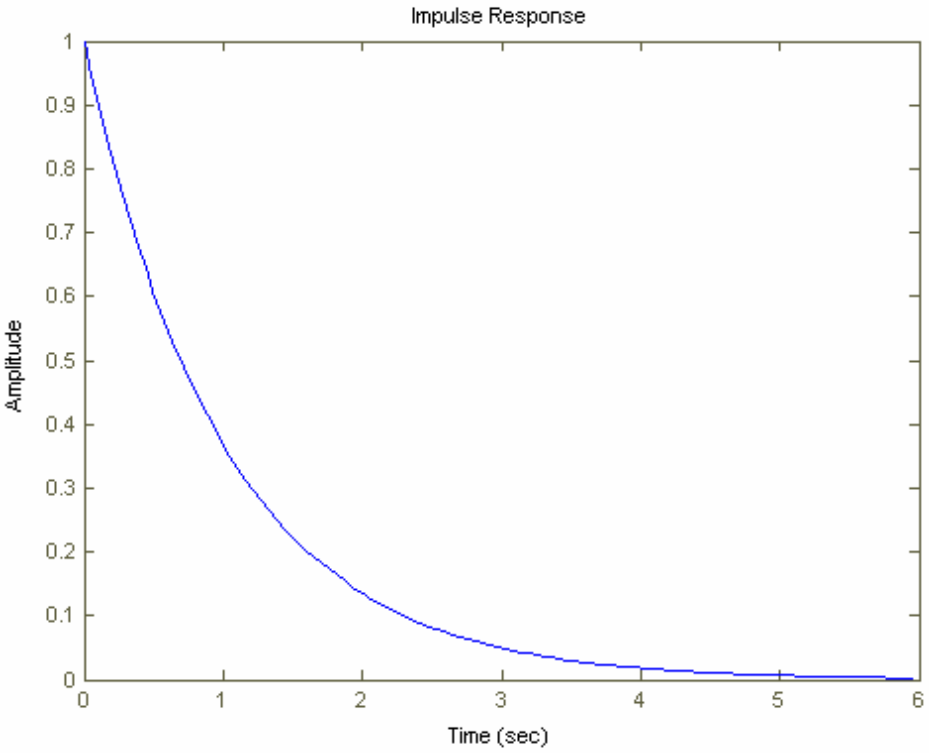


Γράφημα 2.4

impulse(sys)	Υπολογίζεται και σχεδιάζεται η μοναδιαία κρουστική απόκριση του συστήματος με συνάρτηση μεταφοράς sys (sys=*μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς)*

Παράδειγμα:

```
>> impulse(sys)
>>
```



Γράφημα 2.5

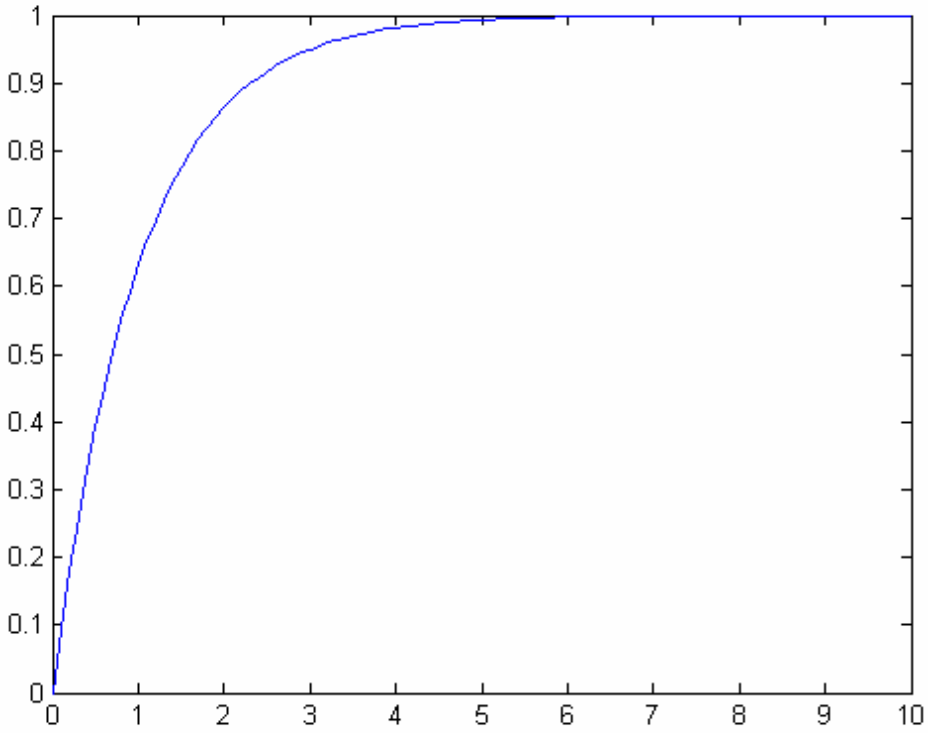
Y = lsim(sys,u,t)	Y = η απόκριση του συστήματος sys στην είσοδο u κατά το χρονικό διάστημα t *υπόδειξη: ορισμός t ως t1:dt:t2, ενώ το u είναι το διάνυσμα τιμών της συνάρτησης εισόδου u = f(t) που υπολογίζεται για το διάνυσμα t*
Y= polyval(p,x)	Y = η τιμή του πολυωνύμου p(r) όταν r = x *(το x μπορεί να είναι και μιγαδικός)*

Παράδειγμα:

```
>> t=0:0.1:10;
>> u=1;
>> u1=polyval(u,t);
>> y=lsim(sys,u1,t)

y =

       0
  0.0952
  0.1813
  0.2592
  0.3297
  0.3935
  0.4512
  0.5034
  0.5507
  0.5934
  0.6321
  0.6671
  0.6988
  0.7275
  0.7534
  0.7769
  0.7981
  0.8173
  0.8347
  0.8504
  0.8647
  0.8775
  0.8892
  0.8997
  0.9093
  0.9179
  0.9257
  0.9328
  0.9392
  0.9450
  0.9502
  0.9550
  0.9592
  0.9631
  0.9666
  0.9698
  0.9727
  0.9753
  0.9776
  0.9798
  0.9817
  0.9834
  0.9850
  0.9864
  0.9877
  0.9889
  0.9899
  0.9909
  0.9918
  0.9926
  0.9933
```

0.9939
0.9945
0.9950
0.9955
0.9959
0.9963
0.9967
0.9970
0.9973
0.9975
0.9978
0.9980
0.9982
0.9983
0.9985
0.9986
0.9988
0.9989
0.9990
0.9991
0.9992
0.9993
0.9993
0.9994
0.9994
0.9995
0.9995
0.9996
0.9996
0.9997
0.9997
0.9997
0.9998
0.9998
0.9998
0.9998
0.9998
0.9998
0.9999
0.9999
0.9999
0.9999
0.9999
0.9999
0.9999
0.9999
0.9999
0.9999
0.9999
1.0000

>> plot(t,y)



Γράφημα 2.6

Sys = series(sys1,sys2)	Sys = *μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς* που προκύπτει από τη σύνδεση σε σειρά των συστημάτων με συναρτήσεις μεταφοράς sys1 και sys2
Sys = parallel(sys1,sys2)	Sys = *μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς* που προκύπτει από την παράλληλη σύνδεση των συστημάτων με συναρτήσεις μεταφοράς sys1 και sys2
Sys = feedback(sys1,sys2)	Sys = *μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς* που προκύπτει από τη σύνδεση των συστημάτων sys1 και sys2 σε σύστημα κλειστού βρόχου, όπου sys1: συνάρτηση εγκατάστασης και sys2: συνάρτηση ανατροφοδότησης (αρνητική ανατροφοδότηση)

Παράδειγμα:

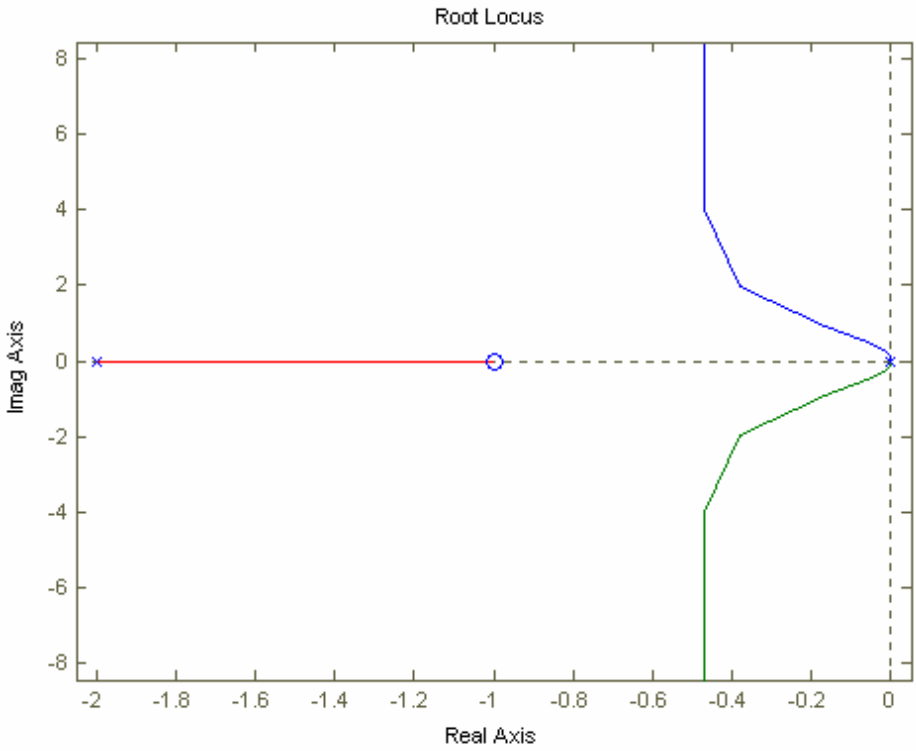
```
>> num=2;
>> den=[2 2];
>> sys=tf(num,den)

Transfer function:
  2
-------
2 s + 2

>> num1=4;
```

```
>> den1=[3 1];
>> sys1=tf(num1,den1)

Transfer function:
  4
-------
3 s + 1

>> Ser=series(sys,sys1)

Transfer function:
      8
---------------
6 s^2 + 8 s + 2

>> par=parallel(sys,sys1)

Transfer function:
  14 s + 10
---------------
6 s^2 + 8 s + 2

>> fed=feedback(sys,sys1)

Transfer function:
   6 s + 2
----------------
6 s^2 + 8 s + 10

>>
```

rlocus(sys)	Υπολογισμός και σχεδίαση του γ.τ.ρ. της εξίσωσης 1+KG(s) = 0, όπου G(s): η συνάρτηση μεταφοράς συστήματος που αναπαρίσταται με τη μεταβλητή-αντικείμενο τύπου συνάρτησης μεταφοράς sys

Παράδειγμα:

```
> num=[4 4];
>> den=[1 2 0 0];
>> sys=tf(num,den)

Transfer function:
 4 s + 4
-----------
s^3 + 2 s^2

>> rlocus(sys)
```



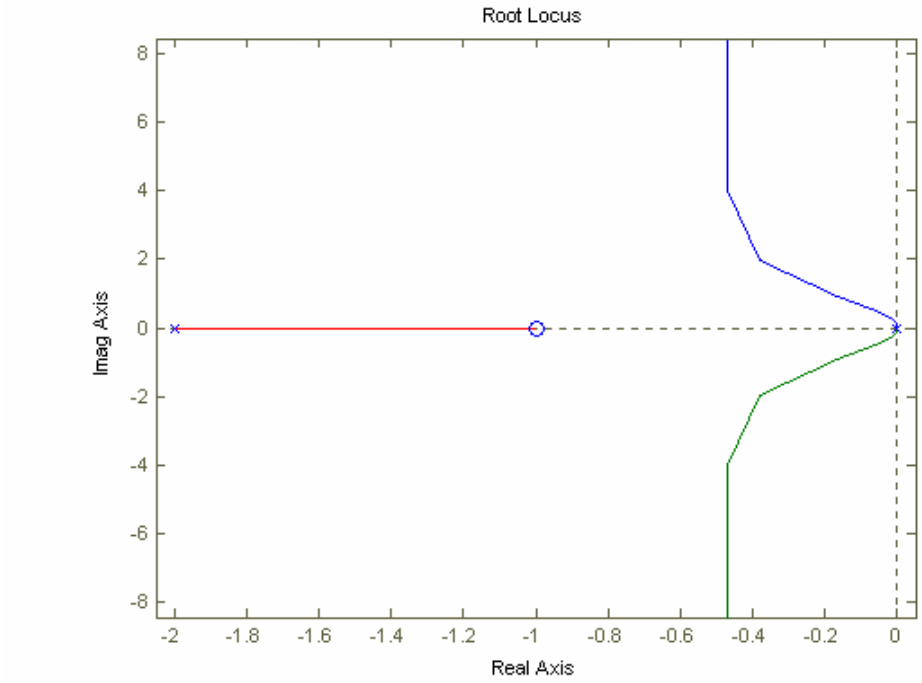
Γράφημα 2.7

[k,poles] = rlocfind(sys)	Εύρεση της τιμής του K και των αντίστοιχων ριζών της χαρακτηριστικής εξίσωσης 1+KG(s) = 0 από το γ.τ.ρ. με γραφικό τρόπο (προϋποθέτει τη σχεδίαση του γ.τ.ρ. μέσω της εντολής rlocus(sys))

Παράδειγμα:

Αρχικά υπολογίζουμε και σχεδιάζουμε το γ.τ.ρ. της εξίσωσης 1+KG(s) = 0.

```
>> num=[4 4];
>> den=[1 2 0 0];
>> sys=tf(num,den)

Transfer function:
 4 s + 4
-------------
s^3 + 2 s^2

>> rlocus(sys)
```

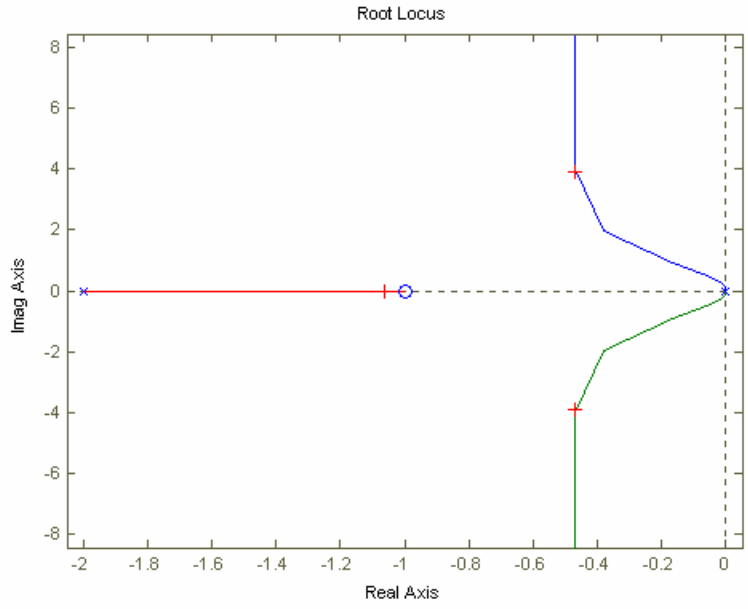


Γράφημα 2.8

Στη συνέχεια χωρίς να κλείσουμε το διάγραμμα γράφουμε στο παράθυρο της MATLAB την εντολή:

```
>> [k,poles]=rlocfind(sys)
Select a point in the graphics window
```

Και επιλέγουμε ένα σημείο στο διάγραμμα:



Γράφημα 2.9

Στο παράθυρο της MATLAB εμφανίζεται το επιλεγμένο σημείο:

```
selected_point =

  -0.4601 + 3.9036i


k =

    4.1130


poles =

  -0.4678 + 3.9035i
  -0.4678 - 3.9035i
  -1.0644

>>
```

[num,den] = ord2(wn,z)	Δημιουργία συνάρτησης μεταφοράς (πολυώνυμα αριθμητή-παρανομαστή: num και den, αντίστοιχα) πρότυπου συστήματος $2^{ης}$ τάξης με επιλεγμένα ζ και ω_n

Παράδειγμα:

```
>> wn=1.932;
>> z=0.69;
>> [num,den]=ord2(wn,z)

num =

     1


den =

    1.0000    2.6662    3.7326

>>
```

[z,p,k] = tf2zp(num,den)	Εύρεση των μηδενικών και πόλων (z και p αντίστοιχα) της συνάρτησης μεταφοράς με πολυώνυμα αριθμητή-παρανομαστή num και den, αντίστοιχα

Παράδειγμα:

```
>> num=[4 4];
>> den=[1 2 0 0];
>> [z,p,k]=tf2zp(num,den)

z =

   -1


p =

    0
    0
   -2


k =

    4

>>
```

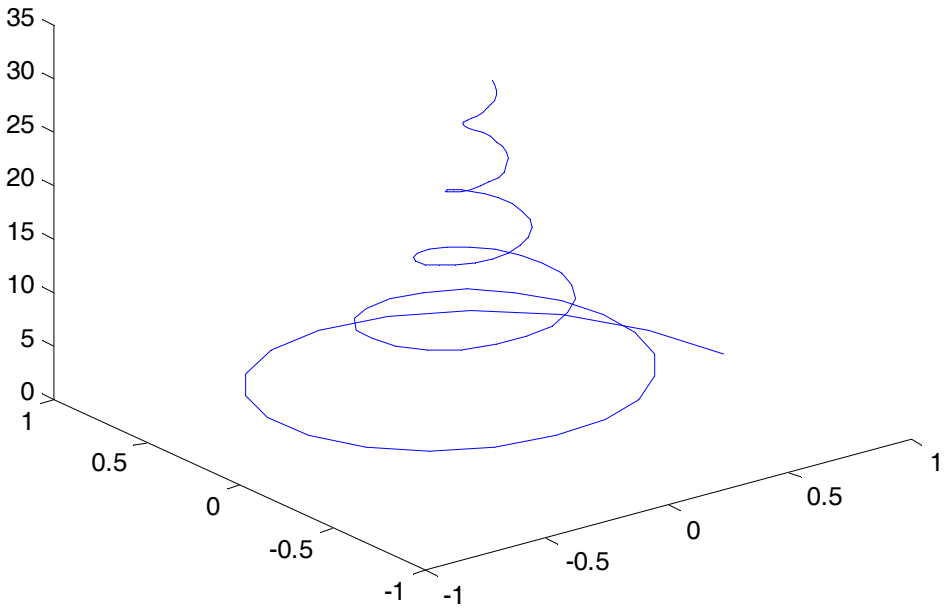
ΚΕΦΑΛΑΙΟ 3
ΓΡΑΦΙΚΑ ΣΤΟ ΧΩΡΟ

3.1. Η συνάρτηση plot3

Όποιος γνωρίζει τη χρήση της συνάρτησης *plot* στο επίπεδο δεν έχει καμία δυσκολία να κατανοήσει τη χρήση της συνάρτησης *plot3*. Η μόνη διαφορά βρίσκεται στο γεγονός ότι τώρα χρησιμοποιείται ένα τρίτο διάνυσμα για να περιγραφεί η τρίτη συνιστώσα των σημείων. Για παράδειγμα, για να ζωγραφίσουμε μια τρισδιάστατη φθίνουσα έλικα μπορούμε να χρησιμοποιήσουμε τις παρακάτω εντολές.

```
>> t=linspace (0,10*pi);
>> a=exp(-t/10);
>> X=a.*cos(t);
>> Y=a.*sin(t);
>> Z=t;
>> plot3(X,Y,Z)
```

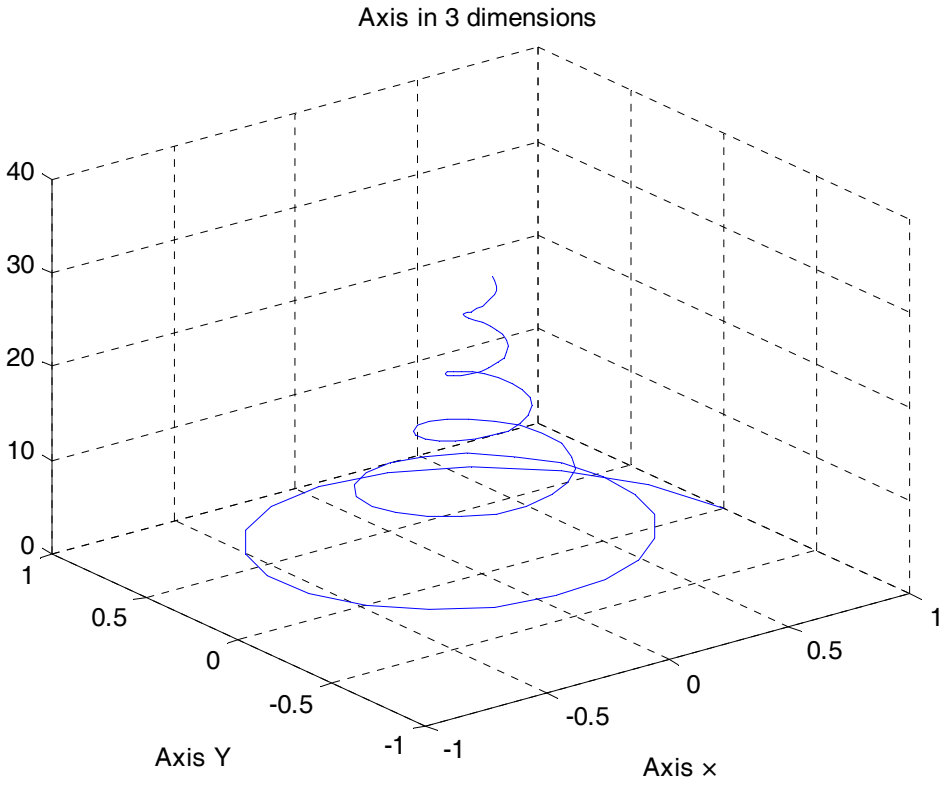
Τα αποτελέσματα φαίνονται στην παρακάτω εικόνα (Γράφημα 3.1)
:



Γράφημα 3.1

Προσέξτε ότι οι άξονες ζωγραφίζονται έτσι ώστε να υπάρχει καλή οπτική εικόνα για την παράσταση. Ο άξονας των Z είναι προφανώς ο κάθετος άξονας. Αν δεν είναι κατανοητό ποιος είναι ο άξονας X και ποιος ο άξονας Y μπορούμε να χρησιμοποιήσουμε τις εντολές *xlabel, ylabel* για να τους δούμε. Φυσικά, στις τρεις διαστάσεις υπάρχει και η συνάρτηση *zlabel*. Βλέποντας το αποτέλεσμα των παρακάτω εντολών (Γράφημα 3.2)

```
>> xlabel('Axis X')
>> ylabel('Axis Y')
>> title('Axis in 3 dimensions')
>> grid on
```

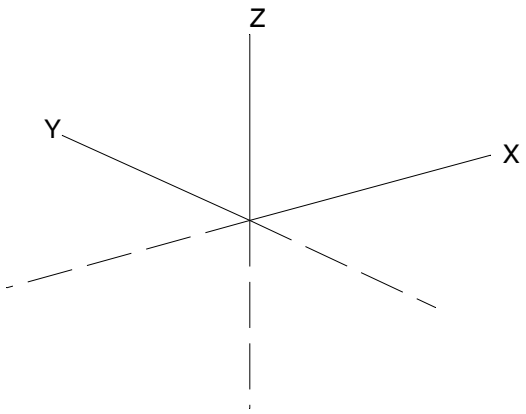


Γράφημα 3.2

στην προηγούμενη εικόνα είναι εύκολο να εντοπίσουμε τους άξονες X και Y και να δούμε ποια είναι τα θετικά τους τμήματα και ποια τα αρνητικά. Η εντολή grid που χρησιμοποιήθηκε βοηθάει να δούμε την αρχή των αξόνων. Για καλύτερη κατανόηση μπορούμε να αφήσουμε το ίδιο το MATLAB να αποκαλύψει τα μυστικά του. Οι παρακάτω εντολές ζωγραφίζουν τους θετικούς άξονες με συνεχείς γραμμές και τους αρνητικούς με διακεκομμένες. Η εντολή *view(3),* που χρησιμοποιείται είναι απαραίτητη για την εμφάνιση τρισδιάστατης παράστασης. Η λειτουργία της θα επεξηγηθεί στο επόμενο τμήμα. Το αποτέλεσμα των εντολών αυτών φαίνεται στην

παρακάτω εικόνα (Γράφημα 3.3). Με την εντολή clf διαγράφεται η παράσταση από το ενεργό παράθυρο. Προσέξτε και την τρισδιάστατη μορφή της συνάρτησης *text*.

```
>> clf, view(3), hold on
>> plot3([0 1],[0 0],[0 0],'-k'), text(1.05,0,0,'X')
>> plot3([0 -1],[0 0],[0 0],'--k')
>> plot3([0 0],[0 1],[0 0],'-k'), text(0,1.1,0,'Y')
>> plot3([0 0],[0 -1],[0 0],'--k')
>> plot3([0 0],[0 0],[0 1],'-k'), text(0,0,1.1,'Z')
>> plot3([0 0],[0 0],[0 -1],'--k')
>> axis off, hold off
```



Γράφημα 3.3

3.2. Παραστάσεις Επιφανειών

Μια από τις πιο ισχυρές λειτουργίες του MATLAB είναι η δυνατότητα εμφάνισης επιφανειών στον τρισδιάστατο χώρο. Οι επιφάνειες μπορούν να φωτιστούν, να σκιαστούν και να εμπλουτιστούν με χρώματα για να τονιστούν ορισμένες ιδιότητες τους. Στο τμήμα αυτό θα παρουσιάσουμε τις βασικότερες συναρτήσεις για απεικόνιση επιφανειών.

3.2.1. Η συνάρτηση mesh και οι παραλλαγές της.

Έστω ότι θέλουμε να κάνουμε γραφική παράσταση της συνάρτησης

$$Z = f(X,Y)$$

η οποία ως γνωστόν παριστάνει μια επιφάνεια στον τρισδιάστατο χώρο. Για να κάνουμε τη γραφική της παράσταση κατασκευάζουμε πρώτα πολλά σημεία (X_i, Y_i) στο επίπεδο X, Y και στη συνέχεια υπολογίζουμε τα σημεία

$$Z_i = f(X_i, Y_i)$$

Το MATLAB διευκολύνει την κατασκευή των σημείων (X_i, Y_i)με τη συνάρτηση *meshgrid*. Η συνάρτηση *meshgrid* παίρνει δύο μεταβλητές εισόδου, τα διανύσματα X και Y και υπολογίζει δυο μεταβλητές εξόδου τις μήτρες X και Y. Ας δούμε πρώτα τις μήτρες X και Y. Θέτωντας

```
>> x=[1 -1 2 -3 4 7];
>> y=[0 2 1 5];
```

Και πληκτρολογώντας την εντολή

```
 >> [X,Y]=meshgrid(x,y)
```

```
X =

    1    -1     2    -3     4     7
    1    -1     2    -3     4     7
    1    -1     2    -3     4     7
    1    -1     2    -3     4     7


Y =

    0     0     0     0     0     0
    2     2     2     2     2     2
    1     1     1     1     1     1
    5     5     5     5     5     5
```

Έτσι κατασκευάζονται *legth(X)* legth(Y) = 6*4=24* σημεία στο επίπεδο. Το στοιχείο $X(i,j)$ είναι η τετμημένη του σημείου (X_i, Y_i) και το στοιχείο $Y(i,j)$ είναι η τεταγμένη του ίδιου σημείου. Χρησιμοποιώντας τις μήτρες X και Y τα σημεία της

επιφάνειας δίνονται με τη μήτρα Z ίσων διαστάσεων με τις X και Y, η οποία υπολογίζεται με τον τύπο

$$Z = f(X,Y)$$

Για παράδειγμα τα σημεία της επιφάνειας

$$Z = \frac{XY\left(X^2 - Y^2\right)}{X^2 + Y^2}$$

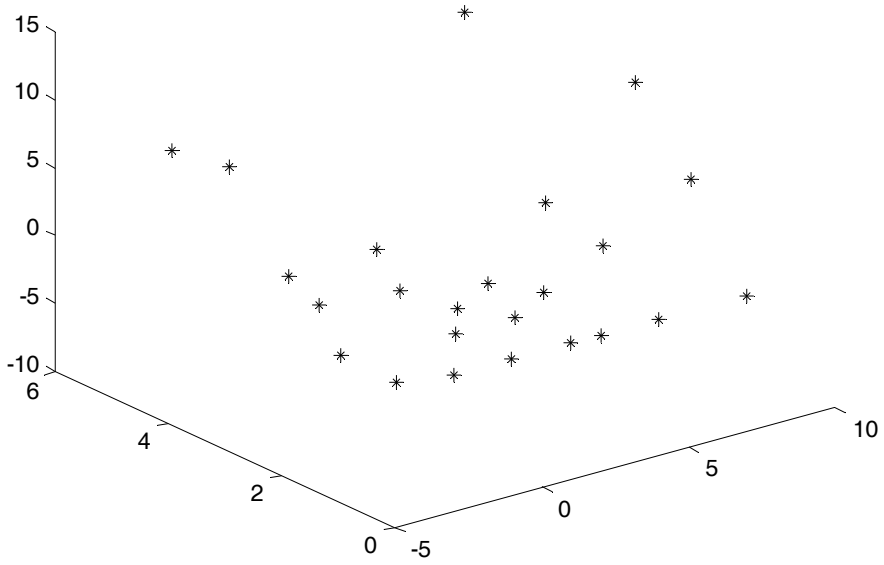
Υπολογίζονται με τη σχέση

```
>> Z=X.*Y.*(X.^2-Y.^2)./(X.^2+Y.^2)

Z =

       0         0         0         0         0         0
  -1.2000    1.2000         0   -2.3077    4.8000   11.8868
       0         0    1.2000   -2.4000    3.5294    6.7200
  -4.6154    4.6154   -7.2414    7.0588   -4.3902   11.3514
```

Παρότι έχουμε πολλά σημεία πάνω στην επιφάνεια και μπορούμε να τα ζωγραφίσουμε με την συνάρτηση

```
plot3(X,Y,Z,'*K')
```

το αποτέλεσμα είναι τελείως απογοητευτικό (Γράφημα 3.4)

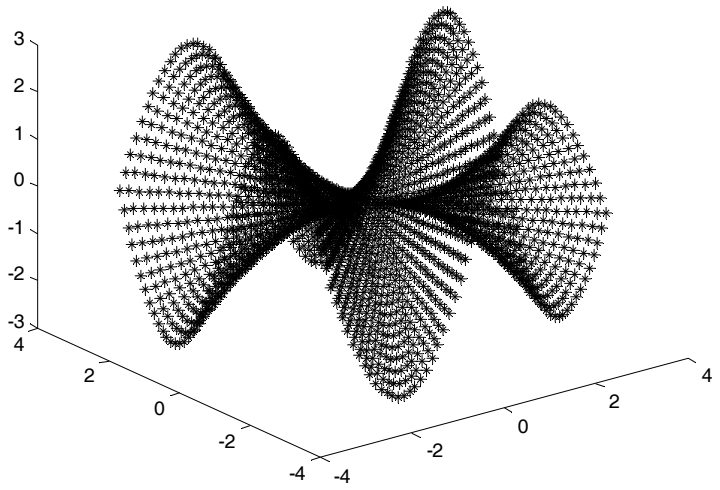


Γράφημα 3.4

Ενώ με τις εντολές

```
>> x=linspace(-3,3,50);
>> y=linspace(-3,3,50);
>> [X,Y]=meshgrid(x,y);
>> Z=X.*Y.*(X.^2-Y.^2)./(X.^2+Y.^2);
>> plot3(X,Y,Z,'*K')
```

Παίρνουμε (Γράφημα 3.5)

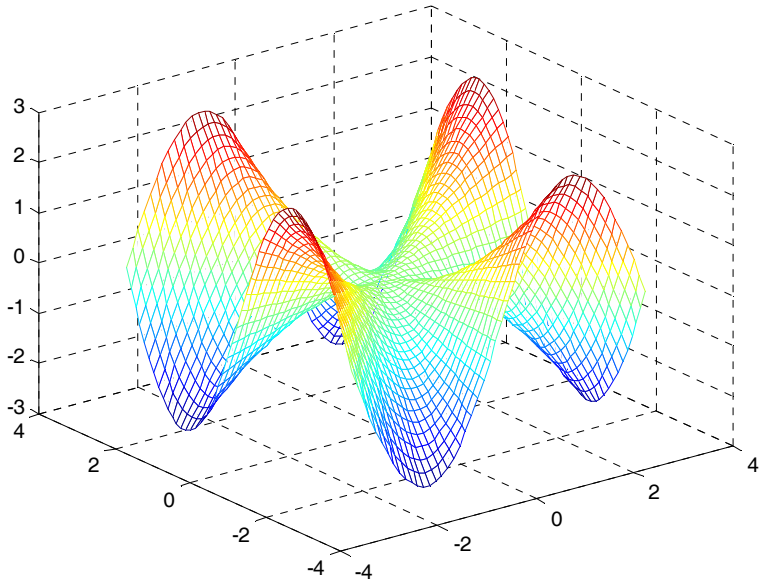


Γράφημα 3.5

Η συνάρτηση

```
>> mesh(X,Y,Z)
```

Εμφανίζει την επιφάνεια (Γράφημα 3.6) χρησιμοποιώντας ένα πλέγμα,



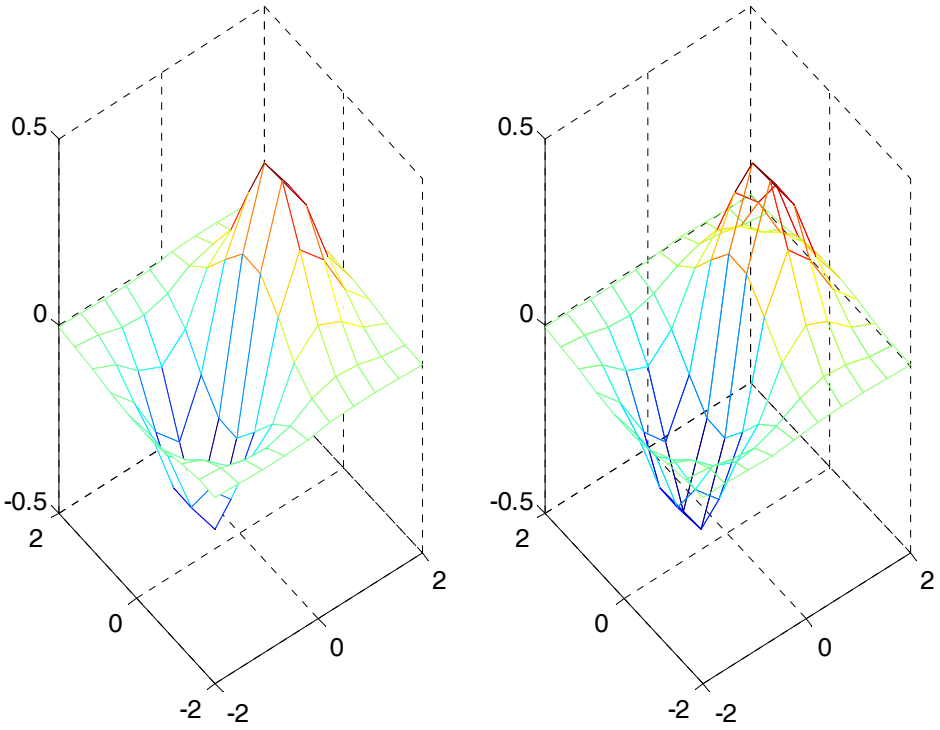
Γράφημα 3.6

Αν θέλουμε να εμφανιστούν τα μη ορατά τμήματα της επιφάνειας χρησιμοποιούμε την εντολή *hidden*, η οποία παίρνει δύο επιλογές *on* και *off*. Σκέτο *hidden* εναλλάσσει από τη μια στην άλλη κατάσταση. Η εξ ορισμού κατάσταση είναι *off*.

Οι επόμενες εντολές ζωγραφίζουν την επιφάνεια (Γράφημα 3.7)

$$Z = xe^{-x^2-y^2}$$

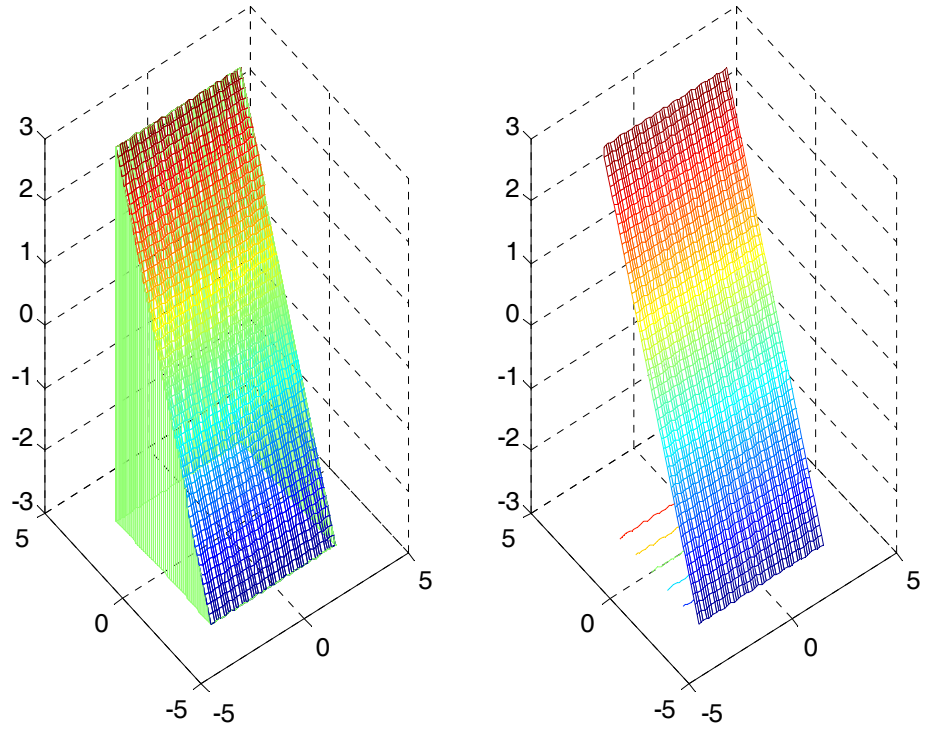
```
>> clf
>> x=linspace(-2,2,10);
>> y=linspace(-2,2,10);
>> [X,Y]=meshgrid(x,y);
>> Z=X.*exp(-X.^2-Y.^2);
>> subplot(1,2,1)
>> mesh(X,Y,Z)
>> hidden on
>> subplot(1,2,2)
>> mesh(X,Y,Z)
>> hidden off
```



Γράφημα 3.7

Οι συναρτήσεις *meshz* και *meshc* είναι παρόμοιες με τη συνάρτηση *mesh*. Η πρώτη κατασκευάζει την επιφάνεια όπως την *mesh* αλλά επιπλέον προσθέτει κατακόρυφες ευθείες (παράλληλες προς τον άξονα Z). Ενώ η δεύτερη προσθέτει ισοσταθμικές καμπύλες στο επίπεδο X, Y. Στη συνέχεια δίνονται παραδείγματα (Γράφημα 3.8) εφαρμογής των συναρτήσεων *meshz* και *meshc* για την επιφάνεια $Z = \frac{-2}{1+X^2+Y^2}$.

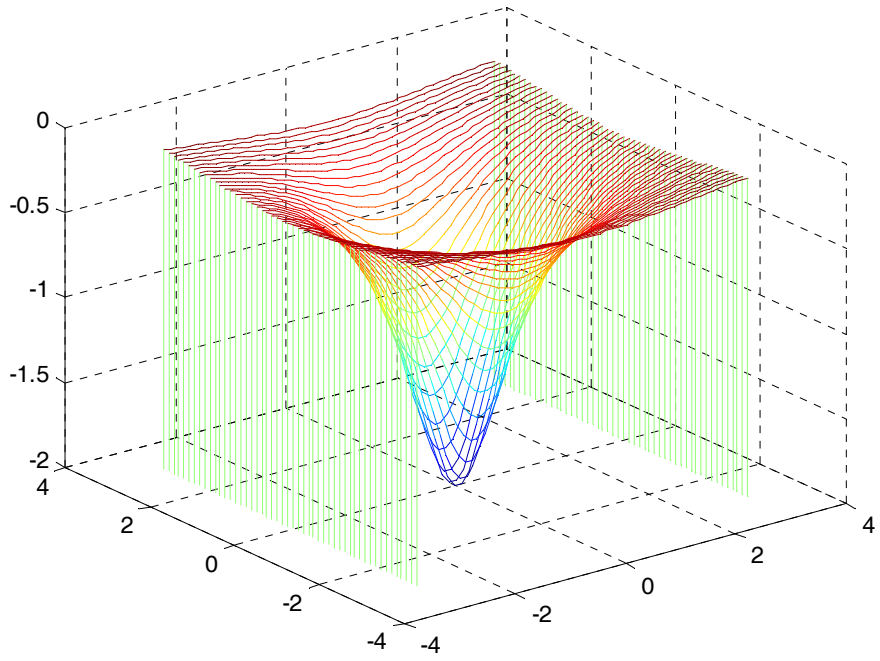
```
>> clf
>> x=linspace(-3,3,50);
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> Z=-2./(1+X.^2+Y.^2);
>> subplot(1,2,1)
>> meshz(X,Y,Y)
>> hidden off
>> subplot(1,2,2)
>> meshc(X,Y,Y)
>> hidden on
```



Γράφημα 3.8

Τέλος υπάρχει και η συνάρτηση *waterfall* η οποία είναι παρόμοια με τη *meshz*. Η συνάρτηση αυτή φαίνεται στο παρακάτω παράδειγμα (Γράφημα 3.9).

```
>> clf
>> waterfall(X,Y,Z)
>> hidden off
```



Γράφημα 3.9

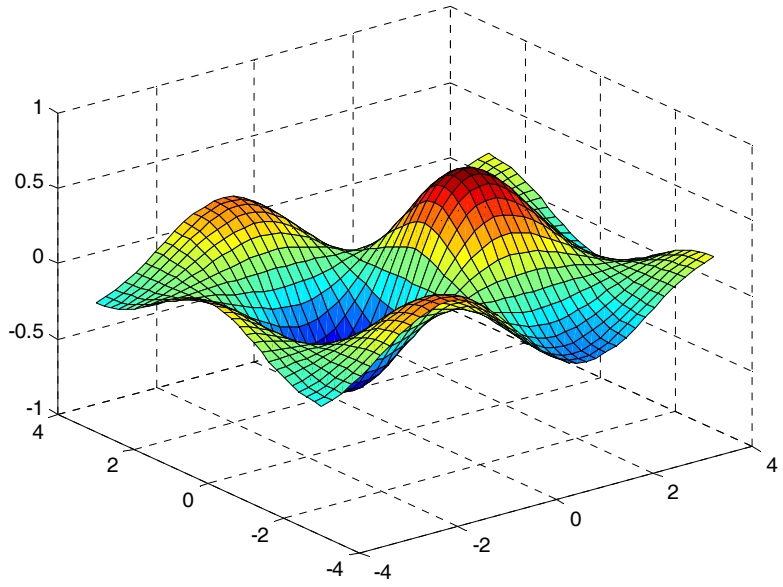
3.2.2. Η συνάρτηση surf και οι παραλλαγές της

Η συνάρτηση *surf* είναι παρόμοια με τη mesh εκτός του ότι τα τμήματα ανάμεσα στις γραμμές πλέγματος (που ονομάζονται μπαλώματα) χρωματίζονται.

Ας δούμε ένα παράδειγμα (Γράφημα 3.10) με τη συνάρτηση surf στην αναπαράσταση της επιφάνειας

$$Z = (\eta\mu X)(\sigma\upsilon\nu X)e^{-\sqrt{\frac{X^2+Y^2}{2}}}$$

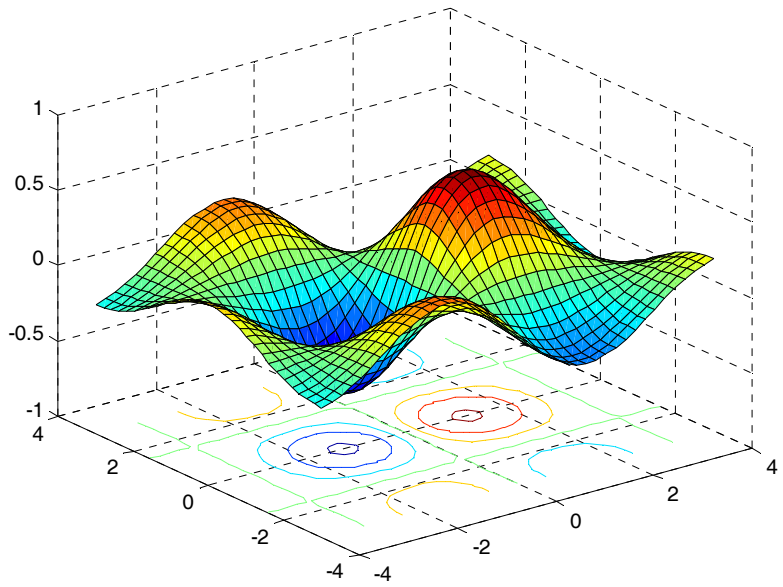
```
>> clf
>> x=-4:.2:4;
>> y=-3:.2:3;
>> [X,Y]=meshgrid(x,y);
>> Z=sin(X).*cos(Y).*exp(-sqrt(X.^2+Y.^2)./3);
>> surf(X,Y,Z)
```



Γράφημα 3.10

Η συνάρτηση *surfc* είναι ίδια με τη συνάρτηση *surf*, διαφέρει μόνο στο ότι προσθέτει ισοβαρείς καμπύλες στο επίπεδο X, Y. Χρησιμοποιώντας την στην προηγούμενη συνάρτηση παίρνουμε (Γράφημα 3.11):

```
>> surfc(X,Y,Z)
```

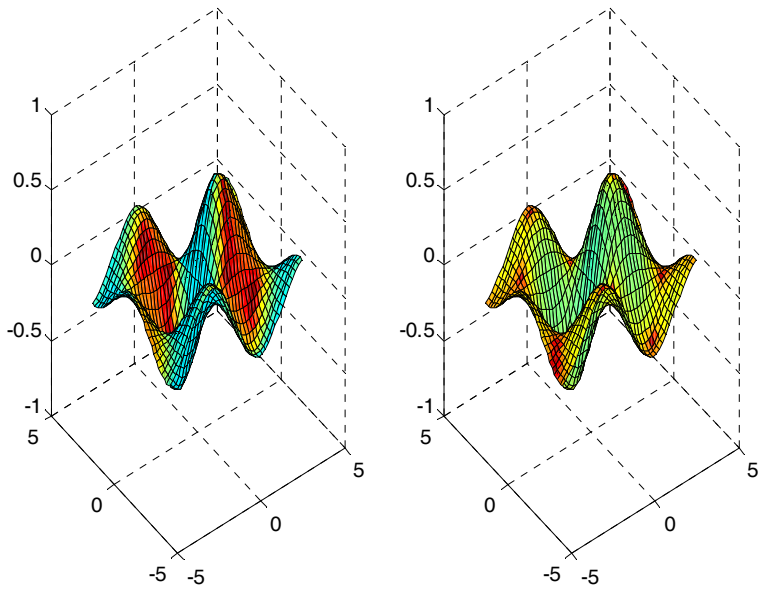


Γράφημα 3.11

Η συνάρτηση *surfl* ζωγραφίζει την επιφάνεια σαν να πέφτει φως πάνω της από μια κατεύθυνση η οποία περνάει από την αρχή των αξόνων και το σημείο *S = [SX SYS Z]*. Η κατεύθυνση του φωτός μπορεί να προσδιοριστεί και σε πολικές συντεταγμένες *S = [αζιμούθιο, ανύψωση]*. Τα μπαλώματα της επιφάνειας χρωματίζονται έτσι ώστε να δείχνουν την ανάκλαση του φωτός. Μπαλώματα κάθετα στην κατεύθυνση του φωτός χρωματίζονται με έντονα κόκκινα χρώματα, ενώ αυτά που είναι παράλληλα στην κατεύθυνση του φωτός έχουν ανοιχτό μπλε χρώμα. Φυσικά υπάρχουν και τα ενδιάμεσα χρώματα. Εκτός από την κατεύθυνση του φωτός μπορούν να προσδιοριστούν και άλλες παράμετροι του, όπως για παράδειγμα το *φως του σκηνικού (ambient light)*, η *διάχυση*, η *ένταση* και η *διασπορά* της ανάκλασης. Οι τελευταίοι τέσσερις παράμετροι προσδιορίζονται με ένα διάνυσμα k. Οι παράμετροι s και k είναι προαιρετικοί. Η εξ' ορισμού κατεύθυνση είναι 45 μοίρες δεξιά της κατεύθυνσης με την οποία βλέπει ο παρατηρητής την εικόνα.

Η χρήση της συνάρτησης *surfl* στην προηγούμενη επιφάνεια (Για δεξιά κατεύθυνση του φωτός κατά 45 μοίρες και για κατακόρυφη κατεύθυνση του φωτός αντίστοιχα) φαίνεται παρακάτω (Γράφημα 3.12).

```
>> subplot(1,2,1)
>> surfl(X,Y,Z)
>> subplot(1,2,2)
>> surfl(X,Y,Z,[0 0 1])
```

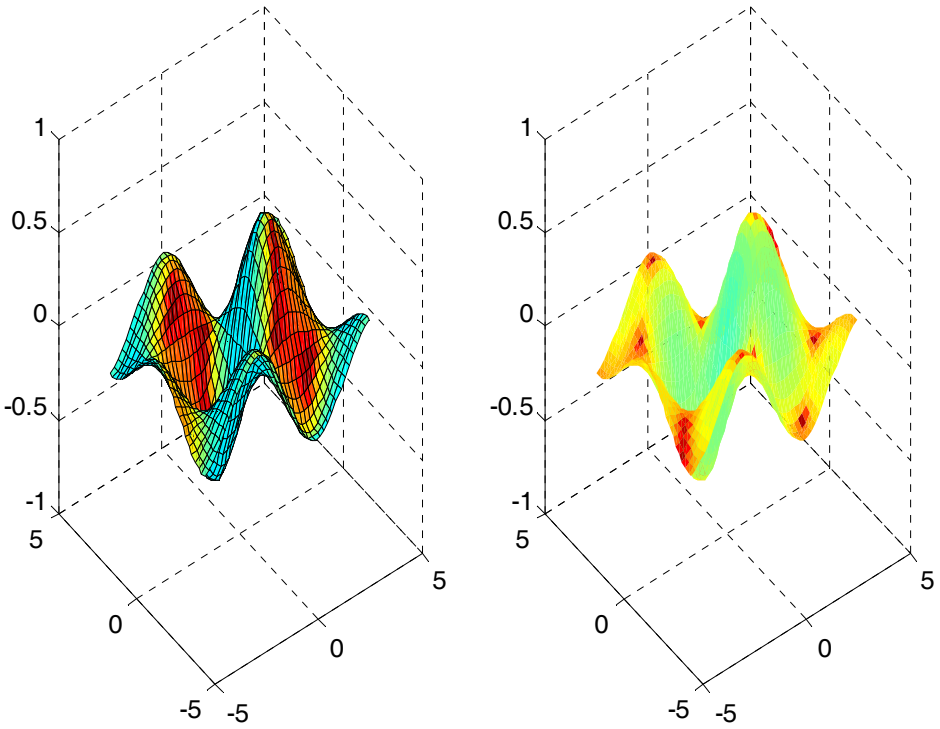


Γράφημα 3.12

Η σκίαση προσδιορίζεται με τη συνάρτηση *shading*. Υπάρχουν τρεις επιλογές, η *επίπεδη (flat)*, η *παρεμβολική (interpolated)*, και η *σκίαση όψεων (facted)*.

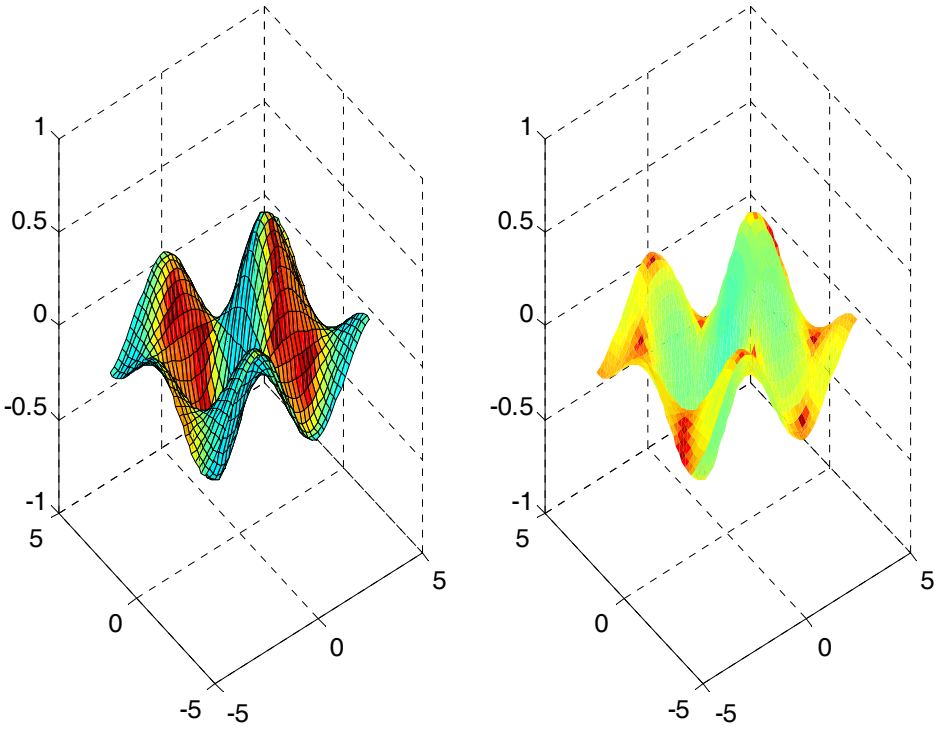
Στην επίπεδη σκίαση κάθε μπάλωμα του πλέγματος έχει σταθερό χρώμα, ενώ οι γραμμές του πλέγματος δεν φαίνονται. Στην παρεμβολική σκίαση ο χρωματισμός αλλάζει ομαλά από το ένα μπάλωμα στο άλλο. Στη σκίαση τριών όψεων είναι εξ' ορισμού επιλογή. Εφαρμόζοντας τη συνάρτηση *shading* στην προηγούμενη επιφάνεια παίρνουμε (Γράφημα 3.13, Γράφημα 3.14):

```
>> shading flat
```



Γράφημα 3.13

```
>> shading interp
```



Γράφημα 3.14

ΚΕΦΑΛΑΙΟ 4
ΚΑΤΑΣΚΕΥΗ ΣΥΝΑΡΤΗΣΕΩΝ ΚΑΙ ΑΡΧΕΙΩΝ ΕΝΤΟΛΩΝ MATLAB

4.1. Γενικά

Εκτός από την εκτέλεση εντολών και την κλήση ενσωματωμένων συναρτήσεων μέσω του παραθύρου εντολών της, η MATLAB παρέχει τη δυνατότητα κλήσης συναρτήσεων και αρχείων εντολών που κατασκευάζονται από το χρήστη. Οι *συναρτήσεις* (functions) και τα *αρχεία εντολών* (scripts) είναι αρχεία κειμένου τα οποία περιέχουν κώδικα MATLAB και χαρακτηρίζονται ως *m-files*, καθώς τα ονόματα τους έχουν την κατάληξη *.m*.

- Τα αρχεία εντολών δεν δέχονται ορίσματα εισόδου και δεν επιστρέφουν ορίσματα εξόδου. Χρησιμοποιούν δεδομένα (μεταβλητές) του χώρου εργασίας.
- Οι συναρτήσεις δέχονται ορίσματα εισόδου και επιστρέφουν ορίσματα εξόδου (όπως και οι ενσωματωμένες συναρτήσεις της MATLAB). Οι μεταβλητές που ορίζονται εντός μιας συνάρτησης είναι *τοπικές* (γνωστές μόνο στη συνάρτηση και όχι στο χώρο εργασίας).

Κάθε m-file, είτε είναι συνάρτηση είτε αρχείο εντολών, δημιουργείται ως αρχείο κειμένου είτε μέσω ενός απλού επεξεργαστή κειμένου (π.χ. notepad) είτε μέσω του MATLAB editor. Για να δημιουργήσουμε ένα νέο m-file μέσω του MATLAB editor, επιλέγουμε το μενού *File* του παραθύρου εντολών και στη συνέχεια *New* → *M-file.* Όταν ολοκληρωθεί η δημιουργία του αρχείου, αυτό αποθηκεύεται με όνομα το οποίο έχει την κατάληξη .m. *Για να είναι γνωστό ένα m-file στο περιβάλλον της MATLAB, θα πρέπει να αποθηκευτεί σε έναν από τους καταλόγους που περιλαμβάνονται στη διαδρομή αναζήτησης της (MATLAB search path).* Η διαδρομή αναζήτησης της MATLAB εμφανίζεται αν δώσουμε την εντολή

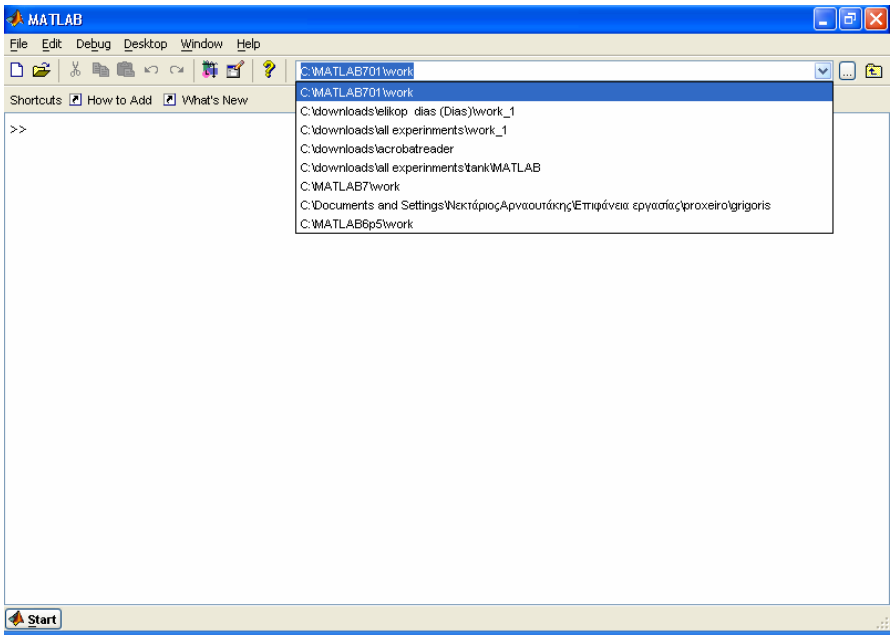
```
» path
```

Κάθε φορά που εκκινείται η MATLAB, ορίζεται ως *τρέχων κατάλογος* (working directory) ένας από τους καταλόγους της διαδρομής αναζήτησης. Στην έκδοση 7.01 ο

κατάλογος αυτός είναι τυπικά ο *x:\MATLAB701\work*, ενώ σε παλαιότερες εκδόσεις ο *x:\MATLAB5\bin*. Ο τρέχων κατάλογος εμφανίζεται εκτελώντας την εντολή

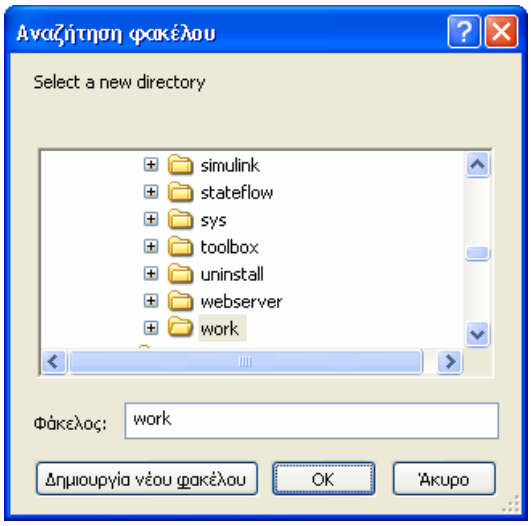
```
» pwd
```

Μια καλή πρακτική για τους νέους χρήστες της MATLAB είναι να αποθηκεύουν τα m-files που δημιουργούν στον εξ' ορισμού τρέχοντα κατάλογο.
Μπορούμε να αλλάξουμε τον εξ' ορισμού κατάλογο αποθήκευσης επιλέγοντας ένα από τους καταλόγους που εμφανίζονται με το πάτημα του βέλους (προς τα κάτω) δίπλα στο όνομα του τρέχοντος καταλόγου που βρίσκεται στην επιφάνεια εργασίας του MATLAB (Εικόνα 4.1).



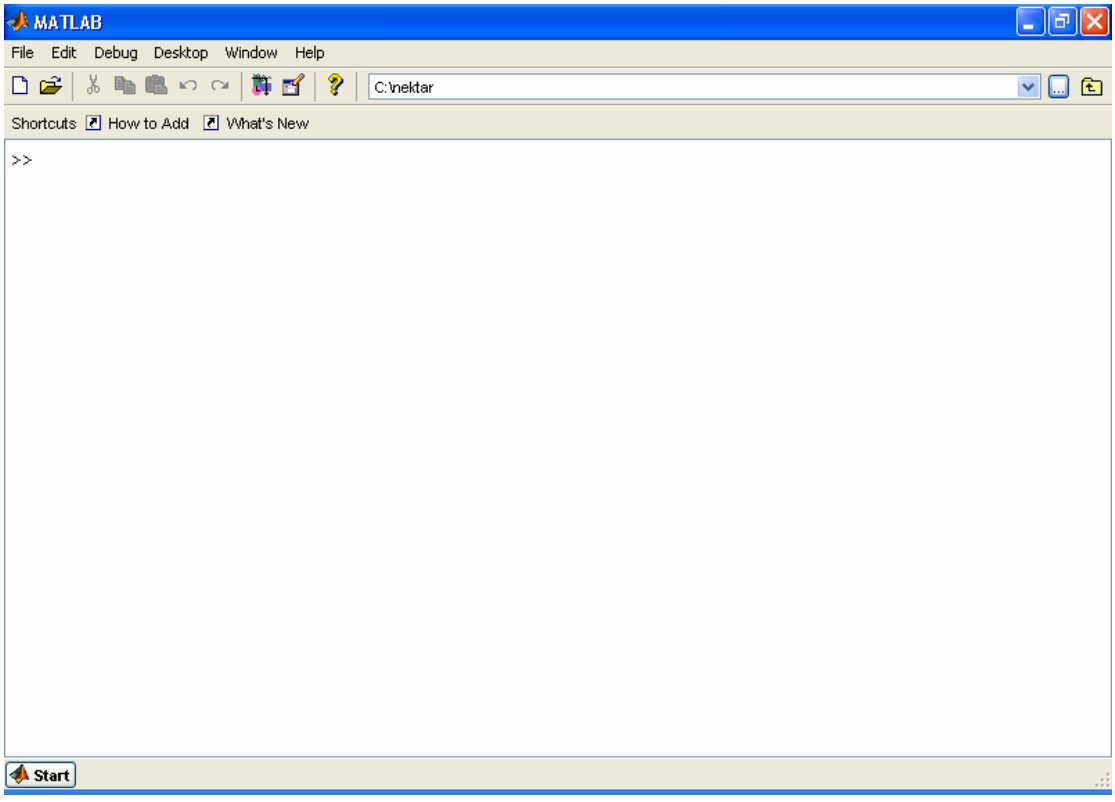
Εικόνα 4.1

Αν επιθυμούμε άλλο κατάλογο επιλέγουμε το κουμπί με τις τρεις τελείες (Εικόνα 4.2).



Εικόνα 4.2

Κάνοντας κλικ πάνω στο κατάλογο και πατώντας το κουμπί *OK* ο κατάλογος γίνεται αυτόματα τρέχον (Εικόνα 4.3)

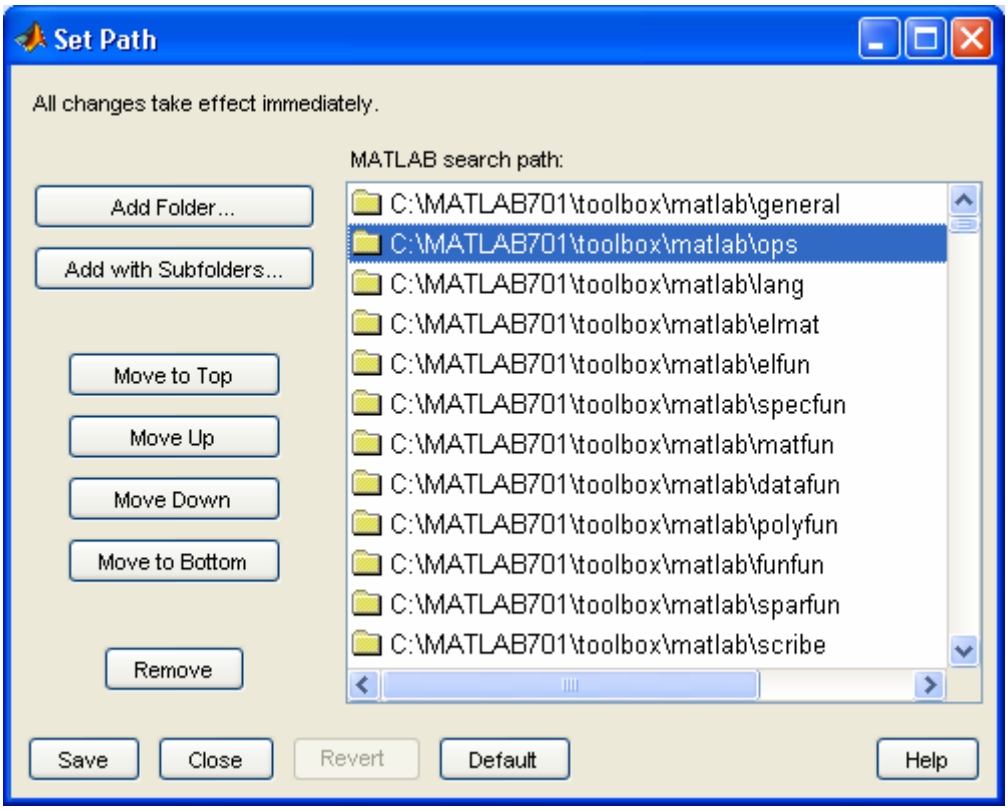


Εικόνα 4.3

Όταν το όνομα ενός εκτελέσιμου αρχείου πληκτρολογηθεί στη γραμμή εντολών, το MATLAB ερευνά τον τρέχοντα κατάλογο και στη συνέχεια όλους τους καταλόγους, τα μονοπάτια των οποίων αναγράφονται σε μια ειδική λίστα (*(MATLAB search*

path)). Αν το όνομα του αρχείου βρεθεί σε ένα από αυτούς τους καταλόγους της λίστας, εκτελείται, διαφορετικά δεν εκτελείται.

Παρέχεται η δυνατότητα να προσθέτουμε, να αφαιρούμε και να αλλάζουμε τη σειρά αναζήτησης στους καταλόγους της λίστας. Με την επιλογή *File – Set path* εμφανίζεται το παράθυρο μονοπατιών (Εικόνα 4.4)



Εικόνα 4.4

Για να προσθέσουμε ένα νέο μονοπάτι καταλόγου πατούμε το κουμπί *Add folder* (προσθήκη καταλόγου) οπότε εμφανίζεται ο γνωστός μας πλοηγός καταλόγων. Μαρκάροντας ένα κατάλογο και πατώντας *OK* το μονοπάτι προστίθεται αυτόματα στη λίστα μονοπατιών και μάλιστα τοποθετείται στην κορυφή της λίστας. Όταν στο παράθυρο μονοπατιών επιλεγεί το κουμπί *Add with subfolders* προστίθεται ο κατάλογος μαζί με τους υποκαταλόγους του, Για να διαγράψουμε ένα μονοπάτι ή να αλλάξουμε τη σειρά του μαρκάρουμε πρώτα τον κατάλογο. Τότε ενεργοποιούμε τα πέντε κουμπιά *Move to top (μετακίνηση στην αρχή), Move up (μετακίνηση μια θέση προς τα επάνω), Remove (διαγραφή), Move down (μετακίνηση μια θέση προς τα κάτω)* και *Move bottom (μετακίνηση στο τέλος)*. Πατώντας το κατάλληλο κουμπί εκτελείται η αντίστοιχη λειτουργία.

Χρειάζεται κάποια προσοχή στην επιλογή των ονομάτων των m-files που δημιουργούνται από το χρήστη, καθώς *αν υπάρχουν δυο m-files με το ίδιο όνομα η MATLAB εκτελεί αυτό που συναντά πρώτο στη διαδρομή αναζήτησης (ξεκινώντας από τον τρέχοντα κατάλογο)*. Τα επιλεγόμενα ονόματα *δεν θα πρέπει να ταυτίζονται με ονόματα ενσωματωμένων συναρτήσεων της MATLAB*.

4.2. Αρχεία εντολών

Όταν καλείται ένα αρχείο εντολών από το παράθυρο εντολών της MATLAB, εκτελούνται οι εντολές που περιέχονται στο αρχείο. Οι εντολές αυτές μπορεί να είναι ορισμοί μεταβλητών, πράξεις, και κλήσεις ενσωματωμένων συναρτήσεων ή συναρτήσεων που έχουν κατασκευαστεί από το χρήστη. Η κλήση ενός αρχείου εντολών γίνεται δίνοντας στο παράθυρο εντολών το όνομα του *χωρίς την κατάληξη .m.* Για παράδειγμα, αν έχει δημιουργηθεί το αρχείο εντολών *my_file.m,* η κλήση του γίνεται με την εντολή

```
» my_file
```

Τα αρχεία εντολών μπορούν να χρησιμοποιούν υπάρχοντα δεδομένα (μεταβλητές) του χώρου εργασίας, ή μπορούν να δημιουργούν και να χρησιμοποιούν νέες μεταβλητές. Παρ' όλο που τα αρχεία εντολών δεν επιστρέφουν ορίσματα εξόδου όπως οι συναρτήσεις, οποιεσδήποτε μεταβλητές που δημιουργούνται από αυτά παραμένουν στο χώρο εργασίας και είναι έτσι διαθέσιμες για περαιτέρω χρήση.

Στη συνέχεια δίνεται ένα παράδειγμα αρχείου εντολών. Οι δύο πρώτες γραμμές είναι γραμμές σχολίων (μια γραμμή σχολίων δηλώνεται από το σύμβολο % στην αρχή της). Η ύπαρξη κενών γραμμών και γραμμών σχολίων επιτρέπεται σε οποιοδήποτε σημείο ενός αρχείου εντολών. Το αρχείο αυτό έχει αποθηκευτεί με το όνομα *my_script.m*. Μέσω αυτού του αρχείου ορίζεται αρχικά ο πίνακας A και στη συνέχεια δημιουργείται ο ανάστροφος του B.

```
% Script example
% Declaration and transposition of a 3x3 matrix

A = [1 2 3;4 5 6;7 8 9];
B = A'
```

Αν δώσουμε την εντολή

» my_script

έχουμε σαν αποτέλεσμα

B =

```
   1   4   7
   2   5   8
   3   6   9
```

Οι μεταβλητές A και B που ορίζονται μέσω του αρχείου εντολών είναι γνωστές στο χώρο εργασίας, όπως διαπιστώνουμε δίνοντας την εντολή

» who

Your variables are:

A B

Οι γραμμές σχολίων που τυχόν περιέχονται στην αρχή ενός αρχείου εντολών, εμφανίζονται αν δώσουμε την εντολή *help όνομα_αρχείου*:

» help my_script

```
 Script example
 Declaration and transposition of a 3x3 matrix
```

Η εντολή *pause* μέσα σε ένα αρχείο εντολών σταματά την εκτέλεση των εντολών. Το MATLAB συνεχίζει την εκτέλεση των επόμενων εντολών μόνο όταν πατηθεί ένα πλήκτρο. Με την επιλογή *pause(t)* το MATLAB σταματά για t δευτερόλεπτα και αρχίζει μετά μόνο του τη λειτουργία.

Για παράδειγμα αν έχουμε ένα αρχείο εντολών που δίνει την περίμετρο και το εμβαδόν ενός τριγώνου

```
perimetros=a+b+c
t=perimetros/2;
emvadon=(t*(t-a)*(t-b)*(t-c))^(1/2)
```

και το εμπλουτίσουμε με τις εντολές *disp, input* που έχουμε δει σε προηγούμενα κεφάλαια και με την εντολή *pause* που είδαμε τώρα παίρνουμε

```
disp(' ')
disp(' Το πρόγραμμα αυτό υπολογίζει την περίμετρο')
disp(' ')
```

```
disp(' και το εμβαδό τριγώνου από τις πλευρές του')
disp(' ')
disp(' Πάτα ένα πλήκτρο να συνεχίσω')
pause
a=input(' Δώσε την πλευρά α ');
b=input(' Δώσε την πλευρά β ');
c=input(' Δώσε την πλευρά γ ');
disp(' Πάτα ένα πλήκτρο να προχωρήσω στους υπολογισμούς ')
pause
perimetros=a+b+c;
disp(' Η περίμετρος είναι :')
disp(perimetros)
t=perimetros/2;
emvathon=(t*(t-a)*(t-b)*(t-c))^(1/2);
disp(' Το εμβαδό είναι :')
disp(emvathon)
```

Εκτελούμε το αρχείο και έχουμε

```
>> triangle_2

 This program calculate the perimeter

 and the area of a triangle from its sides

 press a key for continue
 Type the side  a 5
 Type the side b 6
 Type the side c 7
 Press a key to continue with the calculations
 The perimeter is :
   18

 The area is :
   14.6969
```

4.3. Συναρτήσεις

Οι συναρτήσεις είναι m-files τα οποία περιέχουν εντολές MATLAB, όπως ακριβώς και τα αρχεία εντολών, αλλά έχουν δυο βασικές διαφορές με αυτά: αφ' ενός δέχονται ορίσματα εισόδου και επιστρέφουν ορίσματα εξόδου και αφ' ετέρου οι μεταβλητές που ορίζονται στο εσωτερικό τους δεν είναι γνωστές στο χώρο εργασίας. Η κλήση

μιας συνάρτησης που έχει δημιουργηθεί από το χρήστη γίνεται με τον ίδιο τρόπο που γίνεται η κλήση μιας ενσωματωμένης συνάρτησης της MATLAB (δείτε σχετικά την ενότητα 3 του παρόντος). *Το όνομα της συνάρτησης πρέπει να είναι το ίδιο με το όνομα του m-file στο οποίο είναι αποθηκευμένη*.

Ένα m-file το οποίο ορίζει μια συνάρτηση περιέχει *υποχρεωτικά* στην πρώτη γραμμή του την εντολή (στη γενική της μορφή)

$$function\ [A_1, A_2, ..., A_n] = function\_name(a_1, a_2, ..., a_m)$$

όπου $[A_1, A_2, ..., A_n]$ είναι η λίστα των ορισμάτων εξόδου, *function_name* το όνομα της συνάρτησης και $(a_1, a_2, ..., a_m)$ η λίστα των ορισμάτων εισόδου. Στη συνέχεια δίνεται ο κώδικας της συνάρτησης *my_function.m*, η οποία δέχεται ως όρισμα εισόδου έναν πίνακα και επιστρέφει ως αποτέλεσμα (όρισμα εξόδου) τον ανάστροφο του. Προφανώς η συνάρτηση αυτή κάνει ότι και το αρχείο εντολών *my_script.m*, αλλά για *οποιονδήποτε πίνακα* της μεταβιβάσουμε ως όρισμα εισόδου. Έχει συνεπώς γενικότερη εφαρμογή.

```
function b = my_function(a)
% Function example
% Transposition of a matrix

b = a';
```

Για να εξετάσουμε τον τρόπο λειτουργίας της συνάρτησης, δημιουργούμε στο χώρο εργασίας τον πίνακα A:

```
» A = [1 2 3;4 5 6;7 8 9]

A =

   1   2   3
   4   5   6
   7   8   9
```

Στη συνέχεια καλούμε τη συνάρτηση ως εξής:

```
» B = my_function(A)

B =

   1   4   7
   2   5   8
```

```
   3    6    9
```

Ο πίνακας A μεταβιβάζεται ως όρισμα εισόδου, ενώ στη μεταβλητή B (όρισμα εξόδου) αποθηκεύεται το αποτέλεσμα της κλήσης της συνάρτησης. Οι μεταβλητές a και b που ορίζονται και χρησιμοποιούνται από την συνάρτηση δεν είναι γνωστές στο χώρο εργασίας, όπως μπορούμε να διαπιστώσουμε δίνοντας την εντολή

```
» who
Your variables are:
A       B
```

Οι γραμμές σχολίων που ακολουθούν την πρώτη γραμμή του m-file που ορίζει μια συνάρτηση εμφανίζονται όταν δώσουμε την εντολή *help όνομα_συνάρτησης*, όπως ακριβώς συμβαίνει και με τις ενσωματωμένες συναρτήσεις της MATLAB:

```
» help my_function

  Function example
  Transposition of a matrix
```

4.3.1. Καθολικές μεταβλητές

Στην περίπτωση όπου είναι αναγκαίο μια μεταβλητή να είναι γνωστή ταυτόχρονα σε περισσότερες από μια συναρτήσεις, αυτή δηλώνεται ως *καθολική (global)* σε κάθε μια από αυτές. Εάν είναι επιθυμητό η μεταβλητή αυτή να είναι γνωστή και στο χώρο εργασίας, ή σε αρχείο εντολών το οποίο καλεί τις εν λόγω συναρτήσεις, θα πρέπει να δηλωθεί και εκεί ως καθολική. Η δήλωση μιας μεταβλητής ως καθολικής πρέπει να προηγείται της χρήσης της και γίνεται χρησιμοποιώντας τη δεσμευμένη λέξη *global*, όπως φαίνεται και στο παράδειγμα που ακολουθεί:

```
global G
G = 9.81;
```

Είθισται οι καθολικές μεταβλητές να έχουν ονόματα με κεφαλαία γράμματα ώστε να διακρίνονται από τις υπόλοιπες μεταβλητές, χωρίς αυτό να είναι δεσμευτικό.

4.4. Έλεγχος ροής – λογικοί και συσχετιστικοί τελεστές

Η MATLAB διαθέτει πέντε δομές ελέγχου ροής:

- εντολές if
- εντολές switch
- βρόχους for
- βρόχους while
- εντολές break

4.4.1. Εντολές if

Το γενικό συντακτικό μιας δομής ελέγχου *if* έχει τη μορφή:

if λογική_συνθήκη_1
    μπλοκ_εντολών_1
elseif λογική_συνθήκη_2
    μπλοκ_εντολών_2
……
elseif λογική_συνθήκη_n-1
    μπλοκ_εντολών_n-1
else
    μπλοκ_εντολών_n
end

Η δομή αυτή εξετάζει την ισχύ των λογικών συνθηκών που συνοδεύουν τις δεσμευμένες λέξεις *if* και *elseif*. Η πρώτη από τις λογικές συνθήκες που υπολογίζεται ως *αληθής* (true) συνεπάγεται την εκτέλεση του αντίστοιχου μπλοκ εντολών και την έξοδο από τη δομή *if*. Εαν καμμία λογική συνθήκη δεν είναι αληθής, εκτελείται το μπλοκ εντολών που ακολουθεί το *else*. Οι δομές *elseif* και *else* είναι προαιρετικές, εν αντιθέσει με την ύπαρξη του *end* που σηματοδοτεί την ολοκλήρωση της δομής ελέγχου *if*.

Παράδειγμα:

Να γραφεί μια συνάρτηση με όνομα testif_1. Η συνάρτηση να δέχεται δύο μεταβλητές εισόδου, μια μεταβλητή α, η οποία να παίρνει ακέραιες τιμές και μια μεταβλητή β, η οποία να παίρνει οποιαδήποτε πραγματική τιμή. Η συνάρτηση να υπολογίζει αριθμητικές τιμές ορισμένων πολυωνύμων στην τιμή Χ=β ως εξής.

Αν είναι α=3, να υπολογίζεται η τιμή του πολυωνύμου

$$3X^2 + 2X + 1$$

Αν είναι α=4, να υπολογίζεται η τιμή του πολυωνύμου

$$4X^3 + 3X^2 + 2X + 1$$

Σε κάθε άλλη περίπτωση να υπολογίζεται η τιμή του πολυωνύμου

$$2X + 1$$

Να επαληθευθεί το πρόγραμμα για α=0 και β=-1.

Λύση:
Η συνάρτηση γράφεται:

```
function y=testif_1(a,b)
%Ypologismos ariumhtikon timvn polyionymon
if a==3
   y=polyval([3 2 1],b);
elseif a==4
   y=polyval([4 3 2 1],b);
else
   y=polyval([2 1],b);
end
```

Η επαλήθευση γίνεται πληκτρολογώντας:

```
>> y=testif_1(0,-1)

y =

   -1

>>
```

4.4.2. Εντολές switch

Το γενικό συντακτικό μιας δομής ελέγχου *switch* έχει τη μορφή:

switch μαθηματική_έκφραση
case τιμή_1
μπλοκ_εντολών_1
.....

```
        case τιμή_n
                μπλοκ_εντολών_n
        otherwise
                μπλοκ_εντολών_n+1
end
```

Η δομή ελέγχου *switch* έχει ως αποτέλεσμα την εκτέλεση μπλοκ εντολών ανάλογα με την τιμή της μαθηματικής έκφρασης. Συγκεκριμένα, εκτελείται μόνο το μπλοκ εντολών του πρώτου *case* για το οποίο η τιμή συμπίπτει με την τιμή της έκφρασης. Αν αυτό δεν συμβαίνει για κανένα *case*, εκτελείται το μπλοκ εντολών που ακολουθεί το *otherwise*.

Παράδειγμα:
Χρησιμοποιώντας το προηγούμενο παράδειγμα η λύση γίνεται:

```
function y=testif_2(a,b)
%Ypologismos arithmhtikon timon polionymon
switch a
case 3
   y=3*b^2+2*b+1;
case 4
   y=4*b^3+3*b^2+2*b+1;
otherwise
   y=2*b+1;
end
```

Η επαλήθευση γίνεται πληκτρολογώντας:

```
> y=testif_2(0,-1)

y =

   -1

>>
```

4.4.3. Βρόχοι for

Ένας βρόχος *for* εκτελεί ένα μπλοκ εντολών για ένα καθορισμένο αριθμό επαναλήψεων. Το γενικό συντακτικό μιας δομής ελέγχου *for* έχει τη μορφή:

```
for μετρητής = n1:step:n2
        μπλοκ εντολών
end
```

Ως *μετρητής* χρησιμοποιείται μια μεταβλητή η οποία παίρνει τιμές από *n1* ως *n2* με βήμα *step*. Οι τιμές των *n1*, *n2* και *step* καθορίζουν και το πλήθος των εκτελούμενων επαναλήψεων. Αν η *step* παραληφθεί, θεωρείται ότι έχει την τιμή 1 (σύνηθες). Συνήθως η μεταβλητή *μετρητής* παίρνει ακέραιες τιμές και χρησιμοποιείται εντός του μπλοκ εντολών (π.χ. ως δείκτης γραμμών/στηλών κάποιου πίνακα). Είναι δυνατή η ύπαρξη φωλιασμένων (nested) βρόχων *for*:

for μετρητής_α = n1_α:step_α:n2_α
for μετρητής_β = n1_β:step_β:n2_β
μπλοκ εντολών
end
end

Παράδειγμα:

Να γραφεί μια συνάρτηση με όνομα parag_1. Η συνάρτηση αυτή να υπολογίζει το άθροισμα των τετραγώνων ενός ορισμένου πλήθους αριθμών. Το πλήθος των αριθμών συμβολίζεται με το γράμμα α και δίνεται από το χρήστη.

Λύση:
Η συνάρτηση γράφεται:

```
function s1=parag_1(a)
%Ypologismos athroismatos tetragonon orismenou plithous arithmon
s1=0;
for i=1:1:a
   s1=s1+i^2;
end
```

Η επαλήθευση γίνεται πληκτρολογώντας:

```
>> y=parag_1(9)

y =

  285

>>
```

4.4.4. Βρόχοι while

Ένας βρόχος *while* επαναλαμβάνει το *μπλοκ_εντολών* για ένα ακαθόριστο αριθμό επαναλήψεων εφ' όσον η *λογική_συνθήκη* είναι αληθής. Το γενικό συντακτικό ενός βρόχου *while* έχει τη μορφή:

while λογική_συνθήκη
    μπλοκ_εντολών
end

Εντός του μπλοκ εντολών θα πρέπει να υπάρχει απαραίτητα μια εντολή η οποία επιδρά στην τιμή της λογικής συνθήκης έτσι ώστε αυτή κάποτε να γίνει ψευδής, διαφορετικά ο βρόχος θα γίνει ατέρμονας. Η λογική συνθήκη θα πρέπει να έχει αληθή τιμή πριν την είσοδο του προγράμματος στο βρόχο while.

Παράδειγμα:

Να γραφεί συνάρτηση η οποία να διαιρεί ένα θετικό αριθμό α με ένα άλλο θετικό αριθμό β κάνοντας μόνο προσθέσεις και αφαιρέσεις. Ονομάστε τη συνάρτηση division. Η συνάρτηση να υπολογίζει το πηλίκο και το υπόλοιπο.

Λύση:

Η συνάρτηση γράφεται:

```
function [p,y]=division(a,b)
%Ypologismos pilikou kai ypolipou me prostheseis kai afaireseis
p=0;
y=a;
while y>=b
   y=y-b;
   p=p+1;
end
```

Η επαλήθευση γίνεται πληκτρολογώντας:

```
> [p,y]=division(17,3)

p =

    5


y =

    2
```

>>

4.4.5. Εντολές break

Η εντολή *break* προκαλεί τον πρόωρο τερματισμό ενός βρόχου *for* ή *while* και την έξοδο της ροής του προγράμματος από αυτόν. Σε φωλιασμένους βρόχους, η break προκαλεί την έξοδο μόνο από τον πλέον εσωτερικό βρόχο. Συνήθως η εντολή break περιλαμβάνεται στο μπλοκ εντολών κάποιας δομής ελέγχου *if* εντός βρόχου ανακύκλωσης, ώστε αυτός να τερματιστεί με την ικανοποίηση κάποιας λογικής συνθήκης.

Παράδειγμα:

Η μεταβλητή eps είναι ο μικρότερος θετικός αριθμός τέτοιος ώστε 1+eps ≠ 1. Γνωρίζοντας ότι eps=1/(2^k) να γραφεί ένα αρχείο εντολών (m-file) που να υπολογίζει τον εκθέτη k.

Λύση:

Ξεκινώντας με eps=1 και διαιρώντας τη μεταβλητή eps συνεχώς δια δύο κάποτε η τιμή της θα γίνει τόσο μικρή ώστε να ισχύει στο MATLAB η σχέση 1+eps =1.

Η αρχείο εντολών γράφεται:

```
%arxeio ypologismou ektheth k
eps=1;
for i=1:inf
   eps=eps/2;
   if (1+eps)==1
      break
   end
end
k=i-1
```

Η επαλήθευση γίνεται πληκτρολογώντας:

```
> comp_eps_1

k =

   52

>>
```

Πράγματι:

```
> 2^(-52)
```

```
ans =

  2.2204e-016

>>
```

4.4.6. Λογικοί και συσχετιστικοί τελεστές

Οι λογικές συνθήκες που χρησιμοποιούνται στις δομές ελέγχου *if* και *while* παίρνουν μια από τις τιμές *αληθής* (true) ή *ψευδής* (false) και κατασκευάζονται με τη βοήθεια των *λογικών* (logical) και των *συχετιστικών* (relational) τελεστών. Οι κυριότεροι λογικοί τελεστές της MATLAB είναι οι ακόλουθοι:

Λογικό AND: &

Λογικό OR: |

Λογικό NOT: ~

Παράδειγμα:

1)

```
function y=cal(a,b)
if(a>4) & (a<6)
   y=2*b;
else
   y=3+b;
end
```

Η εκτέλεση της συνάρτησης δίνει:

```
> cal(5,1)

ans =

    2

>>
```

2)

```
 function y=cal(a,b)
if(a>4) | (a<6)
   y=2*b;
else
   y=3+b;
```

```
end
```

Η εκτέλεση της συνάρτησης δίνει:

```
> cal1(10,1)

ans =

    2

>>
```

3)

```
function y=cal(a,b)
if(~a>4) & (~a<6)
   y=2*b;
else
   y=3+b;
end
```

Η εκτέλεση της συνάρτησης δίνει:

```
> cal2(10,1)

ans =

    4

>>
```

Οι κυριότεροι συσχετιστικοί τελεστές της MATLAB είναι οι ακόλουθοι:

- Ισότητας: ==
- Ανισότητας: ~=
- Μικρότερο: <
- Μεγαλύτερο: >
- Μικρότερο ίσο: <=
- Μεγαλύτερο ίσο: >=

Οι συσχετιστικοί τελεστές μπορούν να δεχτούν ως τελεσταίους και διανύσματα ή πίνακες εκτός από βαθμωτές μεταβλητές ή παραστάσεις, αλλά κάτι τέτοιο χρειάζεται προσοχή καθώς η λειτουργία τους είναι διαφορετική στην περίπτωση αυτή. Περισσότερες πληροφορίες για τους λογικούς και συσχετιστικούς τελεστές είναι διαθέσιμες μέσω της βοήθειας της MATLAB (εντολή *help relop*).

2.7. Εκτύπωση και αποθήκευση εικόνων

Η εντολή για την αποθήκευση και εκτύπωση εικόνων είναι η *print*. Θα την διευκρινίσουμε στην εικόνα ενός κανονικού πολυγώνου, το οποίο κατασκευάζεται στο επόμενο παράδειγμα.

Παράδειγμα

Να γράψετε μια συνάρτηση με όνομα *polygono,* η οποία να κατασκευάζει ένα κανονικό πολύγωνο. Η συνάρτηση να δέχεται σαν μεταβλητή εισόδου το πλήθος n των κορυφών, την ακτίνα r του περιγεγραμμένου κύκλου και τη γωνία ph, που σχηματίζεται από τον θετικό άξονα Χ και την ακτίνα του περιγεγραμμένου στο πολύγωνο κύκλου, η οποία καταλήγει στην πρώτη κορυφή. Επιπλέον η συνάρτηση να έχει σαν μεταβλητές εισόδου το χρώμα του εσωτερικού του πολυγώνου και το χρώμα της περιμέτρου του. Το εξ ορισμού χρώμα της περιμέτρου να είναι μαύρο και του εσωτερικού κίτρινο.

Λύση

Χωρίζουμε το διάστημα [0 2π] σε n+1 ισοδιάστατα σημεία. Προσθέτουμε στη συνέχεια τη γωνία ph σε κάθε ένα σημείο. Τα σημεία του κύκλου έχουν συντεταγμένες Χ και Υ, που υπολογίζονται όπως φαίνεται στον επόμενο κώδικα της συνάρτησης.

```
function polygono(n,r,ph,kentro,perimcolor,incolor)
if nargin==4
   perimcolor='-k';
   incolor='y';
end
t=linspace(0,2*pi,n+1)+ph;
y=kentro(1)+r*sin(t);
x=kentro(2)+r*cos(t);
fill(x,y,incolor)
hold on
plot(x,y,perimcolor)
plot(0,0)
axis equal
hold off
```

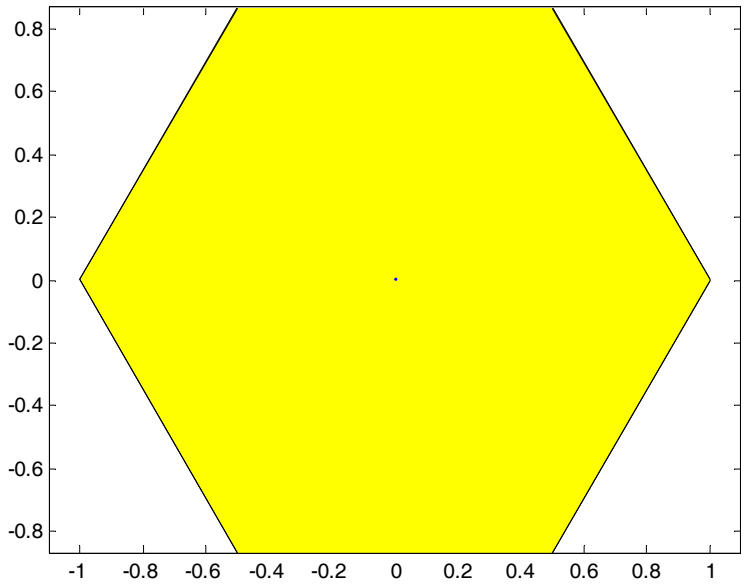
Στον κώδικα χρησιμοποιούμε την εντολή *nargin* για να δώσουμε τις εξ ορισμού τιμές στις μεταβλητές *perimcolor* και *incolor*. Η εντολή *axis equal* εμφανίζει τα κανονικά

πολύγωνα κανονικά. Όταν το πλήθος των κορυφών είναι πολύ μεγάλο, το πολύγωνο φαίνεται σαν κύκλος.

Για να ζωγραφίσουμε ένα εξάγωνο με ακτίνα 1, κέντρο στο σημείο [0, 0] και με μι;α κορυφή πάνω στον άξονα X πληκτρολογούμε

```
>> polygono(6,1,0, [0, 0])
```

Οπότε παίρνουμε την παράσταση της εικόνας 2.10.



Γράφημα 2.10

Αν ο υπολογιστής είναι συνδεδεμένος με ένα εκτυπωτή η εντολή *print*, εκτυπώνει στον εξ' ορισμού εκτυπωτή την παράσταση του ενεργού παραθύρου. Με την ίδια εντολή μπορούμε να αποθηκεύσουμε την εικόνα σε ένα αρχείο. Μια μορφή της εντολής είναι

```
print - devisetype filename
```

Η παράμετρος *devisetype* για *postscript* εκτυπωτές μπορεί να είναι μια από τις επιλογές που φαίνονται στον Πίνακα 2.2

Επιλογή	Ερμηνεία			
PS	Ασπρόμαυρος	Εκτυπωτής	Postcript	
PSC	Έγχρωμος	Εκτυπωτής	Postcript	
PS2	Ασπρόμαυρος	Εκτυπωτής	Postcript	επιπέδου 2
PSC2	Έγχρωμος	Εκτυπωτής	Postcript	επιπέδου 2
eps	Ασπρόμαυρος	Εκτυπωτής	EPSF	
epsc	Έγχρωμος	Εκτυπωτής	EPSF	
eps2	Ασπρόμαυρος	Εκτυπωτής	EPSF	επιπέδου 2
epsc2	Έγχρωμος	Εκτυπωτής	EPSF	επιπέδου 2

Πίνακας 2.2

Το όνομα του αρχείου μπορεί να γραφεί χωρίς την επέκταση. Τότε το MATLAB προσθέτει μόνο του την επέκταση *eps*. Η εικόνα εμφανίζεται μέσα στο MATLAB με την εντολή *open*. Η αποθήκευση της παραπάνω εικόνας και η εν συνεχεία εμφάνιση του αρχείου της μπορεί να γίνει με τις παρακάτω εντολές

```
>> print  -deps2  myfig
>> open('myfig.eps')
```

Η εμφάνιση του αρχείου της εικόνας φαίνεται παρακάτω.

```
%!PS-Adobe-3.0 EPSF-3.0
%%Creator: MATLAB, The Mathworks, Inc.
%%Title: .\myfig.eps
%%CreationDate: 05/20/2005  11:11:18
%%DocumentNeededFonts: Helvetica
%%DocumentProcessColors: Cyan Magenta Yellow Black
%%LanguageLevel: 2
%%Pages: 1
%%BoundingBox:    70   215   543   583
%%EndComments

%%BeginProlog
% MathWorks dictionary
/MathWorks 160 dict begin
% definition operators
/bdef {bind def} bind def
/ldef {load def} bind def
/xdef {exch def} bdef
/xstore {exch store} bdef
% operator abbreviations
/c  /clip ldef
/cc /concat ldef
/cp /closepath ldef
/gr /grestore ldef
/gs /gsave ldef
/mt /moveto ldef
/np /newpath ldef
/cm /currentmatrix ldef
```

```
/sm /setmatrix ldef
/rm /rmoveto ldef
/rl /rlineto ldef
/s {show newpath} bdef
/sc {setcmykcolor} bdef
/sr /setrgbcolor ldef
/sg /setgray ldef
/w /setlinewidth ldef
/j /setlinejoin ldef
/cap /setlinecap ldef
/rc {rectclip} bdef
/rf {rectfill} bdef
% page state control
/pgsv () def
/bpage {/pgsv save def} bdef
/epage {pgsv restore} bdef
/bplot /gsave ldef
/eplot {stroke grestore} bdef
% orientation switch
/portraitMode 0 def /landscapeMode 1 def /rotateMode 2 def
% coordinate system mappings
/dpi2point 0 def
% font control
/FontSize 0 def
/FMS {/FontSize xstore findfont [FontSize 0 0 FontSize neg 0 0]
  makefont setfont} bdef
/ISOLatin1Encoding where {pop /WindowsLatin1Encoding 256 array bdef
ISOLatin1Encoding WindowsLatin1Encoding copy pop
/.notdef/.notdef/quotesinglbase/florin/quotedblbase/ellipsis/dagger
/daggerdbl/circumflex/perthousand/Scaron/guilsinglleft/OE/.notdef/.no
tdef
/.notdef/.notdef/quoteleft/quoteright/quotedblleft/quotedblright/bull
et
/endash/emdash/tilde/trademark/scaron/guilsinglright/oe/.notdef/.notd
ef
/Ydieresis WindowsLatin1Encoding 128 32 getinterval astore pop}
{/WindowsLatin1Encoding StandardEncoding bdef} ifelse
/reencode {exch dup where {pop load} {pop StandardEncoding} ifelse
  exch dup 3 1 roll findfont dup length dict begin
  { 1 index /FID ne {def}{pop pop} ifelse } forall
  /Encoding exch def currentdict end definefont pop} bdef
/isroman {findfont /CharStrings get /Agrave known} bdef
/FMSR {3 1 roll 1 index dup isroman {reencode} {pop pop} ifelse
  exch FMS} bdef
/csm {1 dpi2point div -1 dpi2point div scale neg translate
 dup landscapeMode eq {pop -90 rotate}
  {rotateMode eq {90 rotate} if} ifelse} bdef
% line types: solid, dotted, dashed, dotdash
/SO { [] 0 setdash } bdef
/DO { [.5 dpi2point mul 4 dpi2point mul] 0 setdash } bdef
/DA { [6 dpi2point mul] 0 setdash } bdef
/DD { [.5 dpi2point mul 4 dpi2point mul 6 dpi2point mul 4
  dpi2point mul] 0 setdash } bdef
% macros for lines and objects
/L {lineto stroke} bdef
/MP {3 1 roll moveto 1 sub {rlineto} repeat} bdef
/AP {{rlineto} repeat} bdef
/PDlw -1 def
/W {/PDlw currentlinewidth def setlinewidth} def
/PP {closepath eofill} bdef
/DP {closepath stroke} bdef
```

```
/MR {4 -2 roll moveto dup  0 exch rlineto exch 0 rlineto
  neg 0 exch rlineto closepath} bdef
/FR {MR stroke} bdef
/PR {MR fill} bdef
/L1i {{currentfile picstr readhexstring pop} image} bdef
/tMatrix matrix def
/MakeOval {newpath tMatrix currentmatrix pop translate scale
0 0 1 0 360 arc tMatrix setmatrix} bdef
/FO {MakeOval stroke} bdef
/PO {MakeOval fill} bdef
/PD {currentlinewidth 2 div 0 360 arc fill
   PDlw -1 eq not {PDlw w /PDlw -1 def} if} def
/FA {newpath tMatrix currentmatrix pop translate scale
  0 0 1 5 -2 roll arc tMatrix setmatrix stroke} bdef
/PA {newpath tMatrix currentmatrix pop  translate 0 0 moveto scale
  0 0 1 5 -2 roll arc closepath tMatrix setmatrix fill} bdef
/FAn {newpath tMatrix currentmatrix pop translate scale
  0 0 1 5 -2 roll arcn tMatrix setmatrix stroke} bdef
/PAn {newpath tMatrix currentmatrix pop translate 0 0 moveto scale
  0 0 1 5 -2 roll arcn closepath tMatrix setmatrix fill} bdef
/vradius 0 def /hradius 0 def /lry 0 def
/lrx 0 def /uly 0 def /ulx 0 def /rad 0 def
/MRR {/vradius xdef /hradius xdef /lry xdef /lrx xdef /uly xdef
  /ulx xdef newpath tMatrix currentmatrix pop ulx hradius add uly
  vradius add translate hradius vradius scale 0 0 1 180 270 arc
  tMatrix setmatrix lrx hradius sub uly vradius add translate
  hradius vradius scale 0 0 1 270 360 arc tMatrix setmatrix
  lrx hradius sub lry vradius sub translate hradius vradius scale
  0 0 1 0 90 arc tMatrix setmatrix ulx hradius add lry vradius sub
  translate hradius vradius scale 0 0 1 90 180 arc tMatrix setmatrix
  closepath} bdef
/FRR {MRR stroke } bdef
/PRR {MRR fill } bdef
/MlrRR {/lry xdef /lrx xdef /uly xdef /ulx xdef /rad lry uly sub 2
div def
  newpath tMatrix currentmatrix pop ulx rad add uly rad add translate
  rad rad scale 0 0 1 90 270 arc tMatrix setmatrix lrx rad sub lry
rad
  sub translate rad rad scale 0 0 1 270 90 arc tMatrix setmatrix
  closepath} bdef
/FlrRR {MlrRR stroke } bdef
/PlrRR {MlrRR fill } bdef
/MtbRR {/lry xdef /lrx xdef /uly xdef /ulx xdef /rad lrx ulx sub 2
div def
  newpath tMatrix currentmatrix pop ulx rad add uly rad add translate
  rad rad scale 0 0 1 180 360 arc tMatrix setmatrix lrx rad sub lry
rad
  sub translate rad rad scale 0 0 1 0 180 arc tMatrix setmatrix
  closepath} bdef
/FtbRR {MtbRR stroke } bdef
/PtbRR {MtbRR fill } bdef
/stri 6 array def /dtri 6 array def
/smat 6 array def /dmat 6 array def
/tmat1 6 array def /tmat2 6 array def /dif 3 array def
/asub {/ind2 exch def /ind1 exch def dup dup
  ind1 get exch ind2 get sub exch } bdef
/tri_to_matrix {
  2 0 asub 3 1 asub 4 0 asub 5 1 asub
  dup 0 get exch 1 get 7 -1 roll astore } bdef
/compute_transform {
  dmat dtri tri_to_matrix tmat1 invertmatrix
```

```
  smat stri tri_to_matrix tmat2 concatmatrix } bdef
/ds {stri astore pop} bdef
/dt {dtri astore pop} bdef
/db {2 copy /cols xdef /rows xdef mul dup string
  currentfile
  3 index 0 eq {/ASCIIHexDecode filter}
  {/ASCII85Decode filter 3 index 2 eq {/RunLengthDecode filter} if }
  ifelse exch readstring pop
  /bmap xdef pop pop} bdef
/it {gs np dtri aload pop moveto lineto lineto cp c
  cols rows 8 compute_transform
  {bmap} image gr}bdef
/il {newpath moveto lineto stroke}bdef
currentdict end def
%%EndProlog

%%BeginSetup
MathWorks begin

0 cap

end
%%EndSetup

%%Page: 1 1
%%BeginPageSetup
%%PageBoundingBox:    70   215   543   583
MathWorks begin
bpage
%%EndPageSetup

%%BeginObject: obj1
bplot

/dpi2point 12 def
portraitMode 0216 7344 csm

  629   341  5680  4421 rc
86 dict begin %Colortable dictionary
/c0 { 0.000000 0.000000 0.000000 sr} bdef
/c1 { 1.000000 1.000000 1.000000 sr} bdef
/c2 { 0.900000 0.000000 0.000000 sr} bdef
/c3 { 0.000000 0.820000 0.000000 sr} bdef
/c4 { 0.000000 0.000000 0.800000 sr} bdef
/c5 { 0.910000 0.820000 0.320000 sr} bdef
/c6 { 1.000000 0.260000 0.820000 sr} bdef
/c7 { 0.000000 0.820000 0.820000 sr} bdef
c0
1 j
1 sg
   0    0 6919 5187 rf
6 w
0 4227 5359 0 0 -4227 900 4616 4 MP
PP
-5359 0 0 4227 5359 0 0 -4227 900 4616 5 MP stroke
4 w
DO
SO
6 w
0 sg
 900  389 mt 6259  389 L
```

```
 900 4616 mt 6259 4616 L
6259 4616 mt 6259  389 L
 900 4616 mt  900  389 L
 900 4616 mt 6259 4616 L
 900 4616 mt  900  389 L
1139 4616 mt 1139 4562 L
1139  389 mt 1139  442 L
%%IncludeResource: font Helvetica
/Helvetica /WindowsLatin1Encoding 120 FMSR

1036 4761 mt
(-1) s
1627 4616 mt 1627 4562 L
1627  389 mt 1627  442 L
1474 4761 mt
(-0.8) s
2115 4616 mt 2115 4562 L
2115  389 mt 2115  442 L
1962 4761 mt
(-0.6) s
2603 4616 mt 2603 4562 L
2603  389 mt 2603  442 L
2450 4761 mt
(-0.4) s
3091 4616 mt 3091 4562 L
3091  389 mt 3091  442 L
2938 4761 mt
(-0.2) s
3580 4616 mt 3580 4562 L
3580  389 mt 3580  442 L
3547 4761 mt
(0) s
4068 4616 mt 4068 4562 L
4068  389 mt 4068  442 L
3985 4761 mt
(0.2) s
4556 4616 mt 4556 4562 L
4556  389 mt 4556  442 L
4473 4761 mt
(0.4) s
5044 4616 mt 5044 4562 L
5044  389 mt 5044  442 L
4961 4761 mt
(0.6) s
5532 4616 mt 5532 4562 L
5532  389 mt 5532  442 L
5449 4761 mt
(0.8) s
6020 4616 mt 6020 4562 L
6020  389 mt 6020  442 L
5987 4761 mt
(1) s
 900 4454 mt  953 4454 L
6259 4454 mt 6206 4454 L
 629 4498 mt
(-0.8) s
 900 3966 mt  953 3966 L
6259 3966 mt 6206 3966 L
 629 4010 mt
(-0.6) s
 900 3478 mt  953 3478 L
```

```
6259 3478 mt 6206 3478 L
 629 3522 mt
(-0.4) s
 900 2990 mt  953 2990 L
6259 2990 mt 6206 2990 L
 629 3034 mt
(-0.2) s
 900 2502 mt  953 2502 L
6259 2502 mt 6206 2502 L
 799 2546 mt
(0) s
 900 2014 mt  953 2014 L
6259 2014 mt 6206 2014 L
 699 2058 mt
(0.2) s
 900 1526 mt  953 1526 L
6259 1526 mt 6206 1526 L
 699 1570 mt
(0.4) s
 900 1038 mt  953 1038 L
6259 1038 mt 6206 1038 L
 699 1082 mt
(0.6) s
 900  550 mt  953  550 L
6259  550 mt 6206  550 L
 699  594 mt
(0.8) s
 900  389 mt 6259  389 L
 900 4616 mt 6259 4616 L
6259 4616 mt 6259  389 L
 900 4616 mt  900  389 L
gs 900 389 5360 4228 rc
/c8 { 1.000000 1.000000 0.000000 sr} bdef
c8
1220 -2114 2441 0 1220 2114 -1220 2113 -2441 0 -1220 -2113 6020 2502
7 MP
PP
0 sg
0 0 1220 -2114 2441 0 1220 2114 -1220 2113 -2441 0 -1220 -2113 6020
2502 8 MP stroke
1220 -2114 2441 0 1220 2114 -1220 2113 -2441 0 -1220 -2113 6020 2502
7 MP stroke
3580 2502 PD
gr


end %%Color Dict

eplot
%%EndObject

epage
end

showpage

%%Trailer
%%EOF
```

Η εικόνα επίσης μπορεί να τοποθετηθεί μέσα σ' ένα αρχείο *WORD* επιλέγοντας από το μενού *Edit* του παραθύρου γραφικών *Copy Figure* και στη συνέχεια κάνοντας *paste* στο *WORD* αρχείο.

Αν θέλουμε να αποθηκεύσουμε την εικόνα σε δυαδική μορφή ώστε να εμφανίζεται μέσα στο MATLAB χρησιμοποιούμε την εντολή *save as* από το μενού *File* του παραθύρου γραφικών. Τότε εμφανίζεται ο πλοηγός των *Windows* για την επιλογή του καταλόγου. Αν το αρχείο αποθηκευτεί σε κατάλογο που βρίσκεται στους δρόμους έρευνας του MATLAB, τότε διπλό κλικ του ποντικιού πάνω στο όνομα του αρχείου ανοίγει το παράθυρο γραφικών και εμφανίζεται η εικόνα.

ΚΕΦΑΛΑΙΟ 5
ΤΟ MATLAB ΣΑΝ ΑΝΑΔΡΟΜΙΚΗ ΓΛΩΣΣΑ

5.1. Προγραμματίζοντας αναδρομικά

Μια συνάρτηση του MATLAB καλείται αναδρομική (recursive), αν μέσα στις εντολές της υπάρχει μια τουλάχιστον κλήση στον εαυτό της. Με άλλα λόγια αναδρομική συνάρτηση είναι αυτή η οποία σε κάποια εντολή της περιέχει το όνομα της ίδιας της συνάρτησης. Στο τμήμα αυτό θα περιγράψουμε πως μπορούμε να αξιοποιήσουμε τη δυνατότητα αυτή του MATLAB.

Είναι γνωστό από τα μαθηματικά ότι οι όροι μερικών ακολουθιών και οι τιμές μερικών συναρτήσεων μπορούν να υπολογιστούν αναδρομικά. Ας πάρουμε για παράδειγμα τη συνάρτηση *παραγοντικό (factorial)* n!. Ως γνωστό ο τύπος που δίνει την τιμή του είναι

$$n! = n \cdot (n-1) \cdot (n-2) \ldots 4 \cdot 3 \cdot 2 \cdot 1$$

Το MATLAB έχει μια συνάρτηση για τον υπολογισμό του παραγοντικού την συνάρτηση *factorial*. Η συνάρτηση αυτή όμως δεν είναι αναδρομική, όπως εύκολα μπορούμε να διαπιστώσουμε από τον κώδικα της που εμφανίζεται στην οθόνη με την εντολή *type factorial.*

```
function n = factorial(n)
%FACTORIAL Factorial function.
%   FACTORIAL(N) for scalar N, is the product of all the integers from 1 to N,
%   i.e. prod(1:N). When N is an N-D matrix, FACTORIAL(N) is the factorial for
%   each element of N.  Since double precision numbers only have about
%   15 digits, the answer is only accurate for N <= 21. For larger N,
%   the answer will have the correct order of magnitude, and is accurate for
%   the first 15 digits.
%
%   See also PROD.

%   Copyright 1998-2004 The MathWorks, Inc.
%   $Revision: 1.7.4.6 $   $Date: 2004/06/25 18:52:29 $

N = n(:);
if any(fix(N) ~= N) || any(N < 0) || ~isa(N,'double') || ~isreal(N)
  error('MATLAB:factorial:NNegativeInt', ...
       'N must be a matrix of non-negative integers.')
end
n(N>170) = 171;
m = max([1; n(:)]);
N = [1 1 cumprod(2:m)];
```

n(:) = N(n+1);

Αρχικά ορίζουμε αναδρομικά τη συνάρτηση του παραγοντικού. Θέτοντας *f(n)=n!* μπορούμε να ορίσουμε τη συνάρτηση με τη σχέση

$$f(n) = nf(n-1), \qquad n \geq 2, \qquad f(1) = 1$$

Η σχέση $f(n) = nf(n-1)$ ονομάζεται αναδρομική αφού δεξιά του συμβόλου της ισότητας υπάρχει η ίδια η συνάρτηση.
Γράφουμε συνάρτηση στο MATLAB η οποία υπολογίζει το παραγοντικό αναδρομικά και την οποία την ονομάζουμε *paragontiko*.

```
function a=paragontiko(n)

if n==1
   a=1;
else
    a=n*paragontiko(n-1);
end
```

Οι 6 πρώτες τιμές που δίνει αυτή η συνάρτηση είναι αυτές που παίρνουμε με τον παρακάτω απλό κώδικα.

```
>> for i=1:6
a=paragontiko(i)
end
a =

    1


a =

    2


a =

    6


a =

   24


a =

  120
```

a =

720

5.2. Μια εφαρμογή στο Γραμμικό Πρόβλημα

Για μια τελική επίδειξη των δυνατοτήτων προγραμματισμού που διαθέτει το MATLAB θα παρουσιάσουμε μερικές συναρτήσεις για το γραμμικό πρόβλημα. Με τις συναρτήσεις αυτές θα μπορέσουμε να επιλύσουμε γραμμικά προβλήματα πολύ γενικής μορφής.

5.2.1. Η μαθηματική μορφή του γραμμικού προβλήματος

Στο γραμμικό πρόβλημα ζητείται να βρεθεί το μέγιστο ή το ελάχιστο μιας γραμμικής συνάρτησης των μεταβλητών $X_1, X_2, ..., X_n$, όταν οι μεταβλητές αυτές ικανοποιούν μερικούς γραμμικούς περιορισμούς. Οι περιορισμοί είναι ανισότητες ή ισότητες των αγνώστων μεταβλητών $X_1, X_2, ..., X_n$. Όλες οι μεταβλητές $X_1, X_2, ..., X_n$ παίρνουν μη αρνητικές τιμές. Ένα παράδειγμα γραμμικού προβλήματος είναι το παρακάτω.

$$\min Z = -2X_1 + 3X_2 + 4X_3$$

μ.π.

$$\begin{aligned} &2X_1 - 4X_2 - X_3 \le 8 \\ &-X_2 + 2X_3 = 3 \\ &X_1 - X_2 + 4X_3 \ge -4 \\ &X_j \ge 0, (j = 1,2,3) \end{aligned}$$

Σε μορφή πινάκων το γραμμικό πρόβλημα γίνεται

$$\min Z = c^T X$$

$$\text{Μ.π.} \quad AX \oplus b$$

$$X \ge 0$$

Ή

$$\min\left\{c^T X : AX \oplus b, X \geq 0\right\}$$

Αντικαθιστώντας τα δεδομένα του παραδείγματος έχουμε:

$$c = \begin{bmatrix} -2 \\ 3 \\ 4 \end{bmatrix}, A = \begin{bmatrix} 2 & -4 & -1 \\ 0 & -1 & 2 \\ 1 & -1 & 4 \end{bmatrix}, \oplus = \begin{bmatrix} \leq \\ = \\ \geq \end{bmatrix} \kappa\alpha\iota, b = \begin{bmatrix} 8 \\ 3 \\ -4 \end{bmatrix}$$

Παράδειγμα

Να αναπτυχθεί στο MATLAB ένα πρόγραμμα το οποίο να κατασκευάζει τυχαία γραμμικά προβλήματα της μορφής

$$\min\left\{c^T X : AX \oplus b, X \geq 0\right\}$$

Λύση

Για να κατασκευάσουμε ένα τυχαίο ακέραιο αριθμό στο διάστημα [l, u] κατασκευάζουμε ένα τυχαίο αριθμό α τέτοιο ώστε $0 \leq a \leq 1$.

Κατασκευάζουμε μετά ένα αριθμό X στο διάστημα $[l-0.5, u+0.5]$ χρησιμοποιώντας τη σχέση

$$X = \left(u + 0.5 - \left(l - 0.5\right)\right) \cdot a + \left(l - 0.5\right) = \left(u - l + 1\right) \cdot a + l - 0.5$$

Και μετά στρογγυλοποιούμε τον αριθμό X. Προσέξτε ότι με αυτό τον τρόπο η πιθανότητα να κατασκευαστεί ένας ακραίος αριθμός 1 ή u είναι ίδια με αυτή των ενδιάμεσων αριθμών.

Επομένως ο κώδικας γίνεται

```
function [minmax,c,A,b,eqin]=rglpgenA(m,n,clu,Alu,blu,eqinlu)
% RGLPGENA: Κατασκευάζει τυχαία γραμ. προβλ. τύπου γενικά A
% SYNTAX: [minmax,cT,A,bT,eqinT]=rglpgenA(m,n,clcu,AlAu,blbu,eqinlu)
% Η συνάρτηση κατασκευάζει τυχαία γραμμικά προβλήματα του τύπου
% min/max{c'x:Ax[= <=,=>]b,x>=0}.
% Όλοι οι συντελεστές είναι ακέραιοι και ομοιόμορφα κατανεμημένοι
%-----------------------------------------------------------
%------ Μεταβλητές εισόδου -------------------------------
%-----------------------------------------------------------
```

```
% m = αριθμός περιορισμών (γραμμές του A)
% n = αριθμός μεταβλητών (στήλες του A)
% clu,Alu,blu eqinlu = δισδυάστατα διανύσματα που περιέχουν τα
%                      κάτω και άνω όρια των συντελεστών c, A, b,
%                     eqin αντίστοιχα, π.χ., e.g., clu=[1,9]
%-------------------------------------------------------------
%------ Μεταβλητές εξόδου ---------------------------------
%-------------------------------------------------------------
% minmax=-1(πρόβλημα min), =1(πρόβλημα max)
% c = διάνυσμα συντελεστών κόστους (συντελεστές αντ. συνάρτησης)
% A = μήτρα συντελεστών των τεχνολογικών περιορισμών
% b = διάνυσμα δεξιού μέρους
% eqin(i)=0, 1, 2 (ο περιορισμός i, είναι τύπου =, <=, >=,
αντίστοιχα)
%-----------------------------------------------------------------
% Αν υπάρχουν δυο μεταβλητές εισόδου αυτές είναι m,n.
% Στην περίπτωση αυτή κατασκευάζεται πρόβλημα
% της μορφής min{c'x:Ax<=b,x>=0}, με -9<=c<=-1,0<=A<=9, 1<=b<=9.
%-----------------------------------------------------------------
% Γράφηκε από Παπαρρίζο Κ. στις   23-5-1999
% Τελευταία τροποποίηση στις      27-12-2000
%-----------------------------------------------------------------

if nargin==2
   %Κατασκεύασε ειδικό πρόβλημα επιλέγοντας ειδικές τιμές
   clu=[-9 -1];Alu=[0 9];blu=[1 9];eqinlu=[1 1];
end

% Επέλεξε τυχαία τον τύπο (min ή max) του προβλήματος
minmax=1;
if rand(1)<=0.5,
   minmax=-1;
end
if nargin==2
   %Κατασκεύασε  πρόβλημα ελαχιστοποίησης
   minmax=-1;
end

% Υπολόγισε τυχαία τους συντελεστές των c, A, b
c=rand(1,n);
c=(clu(2)+1-clu(1))*c+clu(1)-0.5;
c=round(c);

A=rand(m,n);
A=(Alu(2)+1-Alu(1))*A+Alu(1)-0.5;
A=round(A);

b=rand(1,m);
b=(blu(2)+1-blu(1))*b+blu(1)-0.5;
b=round(b);

% Επέλεξε τυχαία τους τύπους των περιορισμών
eqin=rand(1,m);
eqin=(eqinlu(2)+1-eqinlu(1))*eqin+eqinlu(1)-0.5;
eqin=round(eqin);
```

Ας δούμε τώρα τη συνάρτηση στην πράξη. Για να κατασκευάσουμε ένα πρόβλημα με 2 τεχνολογικούς περιορισμούς, 5 μεταβλητές, με συντελεστές του διανύσματος κόστους c στο διάνυσμα [-9, -1], συντελεστές πίνακα A στο διάστημα [0, 5], συντελεστές του δεξιού μέρους b στο διάστημα [1, 9] και περιορισμούς της μορφής ≤ πληκτρολογούμε

```
>> [minmax,c,A,b,eqin]=rglpgenA(2,5,[-9 -1],[0 5], [1 9], [1 1])

minmax =

    1


c =

   -5    -4    -2    -1    -3


A =

    1     5     2     0     4
    2     5     5     2     0


b =

    2     2


eqin =

    1     1
```

Βλέπουμε ότι το πρόβλημα που κατασκευάστηκε είναι πρόβλημα μεγιστοποίησης *(minmax=1)*

5.2.2. Η στάνταρ μορφή του γραμμικού προβλήματος

Για εύκολη και κατανοητή περιγραφή των αλγορίθμων τα προβλήματα είναι καλό να παρουσιάζουν ομοιομορφία. Στο γραμμικό πρόβλημα καθιερώθηκε η *στανταρ μορφή (standard form)*, στην οποία όλοι οι τεχνολογικοί περιορισμοί είναι ισότητες και όλες οι μεταβλητές ικανοποιούν τους φυσικούς περιορισμούς, $X \geq 0$. Όταν όλοι οι τεχνολογικοί περιορισμοί είναι ανισότητες το πρόβλημα βρίσκεται στην *κανονική μορφή (canonical form)*.

Ένα πρόβλημα στη γενική μορφή τύπου Α μετατρέπεται στην στάνταρ μορφή με την εισαγωγή των ονομαζόμενων *χαλαρών μεταβλητών (slack variables)*. Έστω Υ η χαλαρή μεταβλητή που εισάγεται στον περιορισμό $a^T X \le b$. Τότε τίθεται $a^T X + Y = b$ και είναι $Y \ge 0$. Επομένως το σύστημα $a^T X \le b$, $X \ge 0$ μετατρέπεται στο ισοδύναμο σύστημα $a^T X + Y \ge 0, X \ge 0, Y \ge 0$. Παρόμοια, αν ο περιορισμός είναι της μορφής $a^T X \ge b$, η εισαγωγή της χαλαρής μεταβλητής Υ τον μετατρέπει στην ισότητα $a^T X - Y = b, Y \ge 0$.

Παράδειγμα.

Να αναπτύξετε μια συνάρτηση με όνομα *genA2stan,* η οποία να μετατρέπει ένα πρόβλημα από τη γενική μορφή στην *στάνταρ* μορφή εισάγοντας κατάλληλες χαλαρές μεταβλητές. Μεταβλητές εισόδου να είναι τα δεδομένα *c, A, eqin* και μεταβλητές εξόδου οι ίδιες οι μεταβλητές. Τα διανύσματα εισόδου να είναι γραμμές ή στήλες ενώ τα διανύσματα εξόδου να είναι γραμμές.

Λύση.

Πρέπει να προσθέσουμε στήλες στον πίνακα Α που θα αντιστοιχούν στους συντελεστές των χαλαρών μεταβλητών και μηδενικά στο διάνυσμα c. Είναι εύκολο να διαπιστωθεί ότι οι συντελεστές της χαλαρής μεταβλητής που εισάγεται στον i περιορισμό είναι e_i, αν ο περιορισμός είναι της μορφής $\le$ και $-e_i$ αν ο περιορισμός είναι της μορφής $\ge$. Εδώ e_i είναι διάνυσμα που έχει παντού μηδενικά εκτός από τη θέση i στην οποία υπάρχει η μονάδα.

Στο τέλος του διανύσματος c θα εισαχθούν τόσα μηδενικά όσες είναι οι χαλαρές μεταβλητές. Με αυτές τις διευκρινίσεις ο επόμενος κώδικας πρέπει να είναι πλήρως κατανοητός.

```
function [c,A,eqin]=genA2stan(c,A,eqin)
% GENA2STAN: Μετασχηματίζει γενικά γραμμικά προβλήματα σε στανταρ
μορφή.
% SYNTAX: [c,A,eqin]=genA2stan(c,A,eqin)
% Η γενική μορφή των γραμμικών προβλημάτων
% είναι min/max{c'x:Ax[= <=,=>]b,x>=0}
% ενώ η στανταρ μορφή είναι min/max{cc'x:AAx=b,x>=0}
% Η συνάρτηση προσαρτά τα απαραίτητα στοιχεία στο διάνυσμα
% κόστους c, τις στήλες των συντελεστών των χαλαρών μεταβλητών
% στη μήτρα Α και υπολογίζει το νέο διάνυσμα eqin.
```

```
%-----------------------------------------------------------
%------ Μεταβλητές εισόδου --------------------------------
%-----------------------------------------------------------
% c = διάνυσμα κόστους
% A = μήτρα συντελεστών του γενικού προβλήματος
% eqin(i)=0, 1, 2 (ο περιορισμός i, είναι τύπου =, <=, >=)
%-----------------------------------------------------------
%------ Μεταβλητές εξόδου ---------------------------------
%-----------------------------------------------------------
% c = νέο διάνυσμα κόστους
% A =  νέα μήτρα συντελεστών
% eqin(i)=0 (επειδή το πρόβλημα εξόδου είναι στη στάνταρ μορφή)
%-----------------------------------------------------------
% Γράφτηκε από Κ. Παπαρρίζο στις 12-6-1999.
% Τελευταία τροποποίηση στις 27-12-2000
%-----------------------------------------------------------

c=c(:);c=c';% Επίτρεψε στο c να είναι στήλη ή γραμμή

[m,n]=size(A);

%Πρόσθεσε μηδενικά στο c και στήλες στη μήτρα A
for i=1:m
   if eqin(i)==1
      ei=zeros(m,1);
      ei(i)=1;
      A=[A ei];
      c=[c 0];
   end
   if eqin(i)==2
      ei=zeros(m,1);
      ei(i)=1;
      A=[A -ei];
      c=[c 0];
   end
end

% Υπολόγισε το νέο διάνυσμα eqin σαν γραμμή
eqin=zeros(1,m);
```

Προσέξτε τη χρήση του συμβόλου : για να επιτραπεί στο διάνυσμα εισόδου *c* να είναι γραμμή ή στήλη. Όσον αφορά στο άλλο διάνυσμα εισόδου *eqin*, αυτό μπορεί να είναι διάνυσμα γραμμή ή στήλη λόγω κώδικα.

Η παρακάτω επιβεβαίωση της ορθότητας του κώδικα δίνει τα αναμενώμενα αποτελέσματα.

```
>> [minmax,c,A,b,eqin]=rglpgenA(3,4)

minmax =

   -1


c =
```

```
   -8   -9   -3   -5

A =

    9    8    6    6
    4    5    8    3
    4    2    0    8

b =

    5    7    4

eqin =

    1    1    1

>> [c,A, eqin]=genA2stan(c,A,eqin)

c =

   -8   -9   -3   -5    0    0    0

A =

    9    8    6    6    1    0    0
    4    5    8    3    0    1    0
    4    2    0    8    0    0    1

eqin =

    0    0    0
```

Επειδή χρησιμοποιούνται συναρτήσεις κατασκευής τυχαίων πινάκων εσείς στον υπολογιστή σας μπορεί να πάρετε διαφορετικά αποτελέσματα.

Τώρα, μπορούμε να περιγράψουμε ένα αλγόριθμό επίλυσης γραμμικών προβλημάτων. Στην επόμενη ενότητα περιγράφεται ο αλγόριθμος *simplex*. Προσέξτε όμως ότι στη μορφή αυτή δεν λύνει όλα τα προβλήματα της *στάνταρ* μορφής.

5.2.3. Ο αλγόριθμος Simplex

Ο αλγόριθμος *Simplex* για την επίλυση γραμμικών προβλημάτων ελαχιστοποίησης στη *στάνταρ* μορφή δουλεύει ως εξής. Διαμερίζει το σύνολο των δεικτών 1:n σε δύο σύνολα δεικτών B και N έτσι ώστε το B να περιέχει m και το N να περιέχει n-m δείκτες. Εδώ n είναι το πλήθος των στηλών και m το πλήθος των γραμμών της

μήτρας A. Το σύνολο B ονομάζεται βασικό σύνολο δεικτών και το N μη βασικό σύνολο.

Σε κάθε επανάληψη ο αλγόριθμος *Simplex* υπολογίζει τα στοιχεία

$$X_B = inv(A(:,B)\cdot b, w = c(B)\cdot invA(:,B)), SN = w\cdot A(:,N).$$

Η πρώτη βάση B επιλέγεται έτσι ώστε η βασική μήτρα $A(:,B)$ να είναι αντιστρέψιμη και το διάνυσμα X_B να είναι ≥ 0. Ο αλγόριθμος δουλεύει έτσι ώστε σε κάθε επανάληψη να είναι $X_B \geq 0$ και η μήτρα $A(:,B)$ να είναι αντιστρέψιμη.

Δοθέντων των στοιχείων X_B , w και SN της τρέχουσας επανάληψης γίνεται προσπάθεια να βρεθούν δύο δείκτες $k \in B$ και $l \in N$. Ο δείκτης l ονομάζεται *εισερχόμενος (entering)* και ο δείκτης *k εξερχόμενος (leaving)*. Η επιλογή των δεικτών γίνεται ως εξής. Ελέγχεται πρώτα αν είναι $SN \geq 0$. Αν είναι, ο αλγόριθμος σταματά και το συμπέρασμα σ' αυτή την περίπτωση είναι ότι το γραμμικό πρόβλημα είναι βέλτιστο. Η βέλτιστη λύση προκύπτει θέτοντας $X(N) = 0$ και $X(B) = XB$. Αν δεν είναι $SN \geq 0$ επιλέγεται δείκτης $l \in N$ τέτοιος ώστε $SN(t) < 0$, όπου t είναι δείκτης τέτοιος ώστε $N(t) = l$. Όταν οι δείκτες l και t έχουν προσδιοριστεί, υπολογίζεται το διάνυσμα στήλη hl από τον τύπο $hl = inv(A(:,B)\cdot A(:,l))$. Αν είναι $hl \geq 0$, ο αλγόριθμος σταματά. Σ' αυτή την περίπτωση το πρόβλημα είναι απεριόριστο. Αν δεν είναι $hl \geq 0$, προσδιορίζεται ένας δείκτης r από τη σχέση

$$\frac{hl(r)}{XB(r)} = \max\left\{\frac{hl(i)}{XB(i)} : l \leq i \leq m\right\}$$

Ο προσδιορισμός του δείκτη r λέμε ότι γίνεται με τον έλεγχο *ελαχίστου λόγου (minimum ratio test)*. Όταν ο δείκτης r προσδιοριστεί, προσδιορίζεται ο δείκτης k από τη σχέση $k = B(r)$. Οι δείκτες k και l εναλλάσσονται κατασκευάζοντας έτσι τη νέα βασική διαμέριση B, N. Η νέα διαμέριση κατασκευάζεται εύκολα θέτοντας $N(t) = k$ και $B(r) = l$. Έχοντας τώρα τη νέα διαμέριση στα χέρια μας μια νέα επανάληψη μπορεί να αρχίσει.

Παράδειγμα

Να αναπτυχθεί μια συνάρτηση με όνομα *psfeasedu,* η οποία ν υλοποίει τον αλγόριθμο *Simplex* . Σαν μεταβλητές εισόδου να χρησιμοποιηθούν τα δεδομένα *c,A,b,* το σύνολο των αρχικών βασικών δεικτών B και ο επιτρεπόμενος μέγιστος αριθμός επαναλήψεων *maxiter*. Ο αλγόριθμος αυτός χρησιμοποιείται για λόγους ασφαλείας. Έτσι αν ξεπεραστεί, ο αλγόριθμος σταματά. Σαν μεταβλητές εξόδου να χρησιμοποιηθούν τα αποτελέσματα *x,w,s,* η τελική αντικειμενική τιμή *zvalue* και μια μεταβλητή *adarap*, οι τιμές της οποίας προσδιορίζουν αν το πρόβλημα είναι βέλτιστό *(adarap=0),* απεριόριστο *(adarap=1)* ή αδύνατο *(adarap=-1).* Η τιμή *adarap=-2* δηλώνει ότι το πρόβλημα δεν έχει λύθει.
Η συνάρτηση να εμφανίζει τα αποτελέσματα σε κάθε επανάληψη. Επίσης οι τιμές *B* και *maxiter* να είναι προεραιτικές. Αν δεν εισάγονται, να τίθεται $B = [n-m+1:n]$ και *maxiter = 10000*.

Λύση

Ο παρακάτω κώδικας υλοποιεί τον αλγόριθμο *Simplex,* όπως ακριβώς περιγράφτηκε προηγουμένως.

```
function [adarap,zvalue,x,w,s,B,niter]=psfeasedu(c,A,b,B,maxiter)
% PSFEASEDU: Πρωτεύων αλγόριθμος simplex
% SYNTAX: [adarap,zvalue,x,w,s,B,niter]=psfeasedu(cT,A,bT,B,maxiter)
% Επιλύει το γραμμικό πρόβλημα στάνταρ μορφής min{c'x:Ax=b,x>=0}
% με τον πρωτεύοντα αλγόριθμο simplex. Η αρχική βάση B πρέπει να
% είναι εφικτή. Ο αλγόριθμος σταματά αν ο μέγιστος αριθμός
% επαναλήψεων, maxiter, ξεπεραστεί.
%-------------------------------------------------------------
%------    Μεταβλητές εισόδου    --------------------------
%-------------------------------------------------------------
% c = διάνυσμα κόστους,  A = μήτρα συντελεστών,
% b = δεξιό μέρος, B = δείκτες αρχικής βάσης
% maxiter = μέγιστος αριθμός επιτρεπόμενων επαναλήψεων.
%-------------------------------------------------------------
%------    Μεταβλητές εξόδου      --------------------------
%-------------------------------------------------------------
% niter = αριθμός εκτελεσθέντων επαναλήψεων
% x,w,s = μεταβλητές απόφασης, δυϊκές και δυϊκές χαλαρές αντίστοιχα
% B = δείκτες τελικής βάσης
% zvalue = τελική τιμή αντικεθμενικής συνάρτησης
% adarap =-1 πρόβλημα αδύνατο
%        =0  πρόβλημα βέλτιστο
%        =1  πρόβλημα απεριόριστο
%        =2  πρόβλημα αδύνατο ή απεριόριστο
```

```
%        =-2 το πρόβλημα δεν έχει λυθεί
%-------------------------------------------------------------
% Γράφτηκε από Παπαρρόζο Κ. στις 31-10-1999.
% Τελευταία τροποποίηση στις 28-12-1999
%-------------------------------------------------------------

[m n]=size(A);

% Αν χρειάζεται, θέσε default τιμές στις μεταβλητές B, maxiter
if nargin==4
   maxiter=10000;
end
if nargin==3
   maxiter=10000;B=[n-m+1:n];
end

% Επίτρεψε στα διανύσματα εισόδου να είναι στήλες ή γραμμές
c=c(:)';b=b(:);

% Αρχικοποιήσεις
adarap=-2;niter=0,B=B
Binv=inv(A(:,B))
xB=Binv*b;xBT=xB'
w=c(B)*Binv
N=setdiff(1:n,B),sN=c(N)-w*A(:,N)

% Κάνε έλεγχο αν η αρχική βάση είναι εφικτή
if any(xB<0)
   error('Η αρχική βάση δεν είναι εφικτή')
end

% Έλεγχος βελτιστότητας και υπέρβασης αριθμού επαναλήψεων
while (any(sN<0))&(niter<=maxiter)

   % Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής
   [sl,t]=min(sN),l=N(t)
   hl=Binv*A(:,l); hlT=hl'
   if hl<=0
      adarap=1; %Πρόβλημα απεριόριστο
      break
   end
   [mrt,r]=max(hl./xB),k=B(r(1))

    % Ανανεώσεις
    niter=niter+1,
    B(r)=l
    Binv=inv(A(:,B))
    xB=Binv*b;xBT=xB'
    w=c(B)*Binv,
    N(t)=k,sN=c(N)-w*A(:,N)
end

% Τελικά αποτελέσματα
if adarap==-2
   adarap=0; %Πρόβλημα βέλτιστο
end
x=zeros(1,n);
x(B)=xB;% Πρόσεξε αυτή την εντολή (γραμμή=στήλη)
s=zeros(1,n);s(N)=sN;
```

```
zvalue=w*b;

%Έλεγξε αν ο μέγιστος αριθμός επαναλήψεων ξεπεράστηκε.
if niter>maxiter
   error('Ο μέγιστος αριθμός επαναλήψεων ξεπεράστηκε.')
end
```

Για να εφαρμόσουμε τη συνάρτηση *psfeasedu* κατασκευάζουμε πρώτα γραμμικά προβλήματα, τα οποία όταν μετατραπούν στη στάνταρ μορφή να διαθέτουν μια αρχική εφικτή βάση B. Τα προβλήματα αυτά έχουν τεχνολογικούς περιορισμούς της μορφής $Ax \le b$, όπου $b \ge 0$ και κατασκευάζονται με τη συνάρτηση

```
>> [minmax,c,A,b,eqin]=rglpgenA(2,5)

minmax =

   -1


c =

   -7   -4   -5   -1   -3


A =

    4    8    6    9    1
    0    4    7    7    4


b =

    9    9


eqin =

    1    1
```

Μετατρέπουμε τώρα το πρόβλημα στην *στάνταρ* μορφή με τη συνάρτηση

```
>> [c1,A1,eqin1]=genA2stan(c,A,eqin)

c1 =

   -7   -4   -5   -1   -3    0    0


A1 =

    4    8    6    9    1    1    0
    0    4    7    7    4    0    1
```

```
eqin1 =

     0     0
```

Τώρα λύνουμε το πρόβλημα με τη συνάρτηση

```
>> [adarap,zvalue,x,w,s,B,niter]=psfeasedu(c1,A1,b)

niter =

     0


B =

     6     7


Binv =

     1     0
     0     1


xBT =

     9     9


w =

     0     0


N =

     1     2     3     4     5


sN =

    -7    -4    -5    -1    -3


sl =

    -7


t =

     1


l =

     1


hlT =
```

```
     4     0

mrt =

    0.4444

r =

     1

k =

     6

niter =

     1

B =

     1     7

Binv =

    0.2500         0
         0    1.0000

xBT =

    2.2500    9.0000

w =

   -1.7500         0

N =

     6     2     3     4     5

sN =

    1.7500   10.0000    5.5000   14.7500   -1.2500

sl =

   -1.2500
```

t =

5

l =

5

hlT =

0.2500 4.0000

mrt =

0.4444

r =

2

k =

7

niter =

2

B =

1 5

Binv =

0.2500 -0.0625
0 0.2500

xBT =

1.6875 2.2500

w =

-1.7500 -0.3125

N =

6 2 3 4 7

```
sN =

    1.7500   11.2500    7.6875   16.9375    0.3125

adarap =

     0

zvalue =

  -18.5625

x =

    1.6875         0         0         0    2.2500         0         0

w =

   -1.7500   -0.3125

s =

         0   11.2500    7.6875   16.9375         0    1.7500    0.3125

B =

     1     5

niter =

     2
```

Το ελληνικό ερωτηματικό χρησιμοποιήθηκε για να μην εμφανιστούν τα τελικά αποτελέσματα δυο φορές.

Βλέπουμε λοιπόν ότι το πρόβλημα λύθηκε σε 2 επαναλήψεις και ότι το πρόβλημα είναι βέλτιστο.

5.2.4. Γραφική επίλυση γραμμικού προβλήματος

Πρώτα θα λύσουμε το πρόβλημα αλγεβρικά μέσα από το παράδειγμα 5.2.1.

Παράδειγμα 5.2.1

Να γραφεί συνάρτηση με όνομα *solverLPinR2* για την επίλυση του γραμμικού προβλήματος στο επίπεδο. Δεδομένα εισόδου να είναι τα στοιχεία *minmax, c, A, b* και *eqin*. Η συνάρτηση να υπολογίζει τα παρακάτω στοιχεία.

1) Τη μεταβλητή *adarap*, οι τιμές της οποίας δηλώνουν αν το πρόβλημα είναι αδύνατο (-1), βέλτιστο (0) ή απεριόριστο (1).
2) Την βέλτιστη αντικειμενική τιμή *zopt* και τη βέλτιστη κορυφή *xopt*. Στην περίπτωση που το πρόβλημα δεν είναι βέλτιστο να τίθεται *zopt=0* και *xopt=[]*.
3) Τα σημεία που είναι τομές ευθειών χωρισμένα σε εφικτά και μη εφικτά σημεία. Τα σημεία να είναι στήλες των μητρών *Pfeas* και *Pinfeas*. Επίσης να υπολογίζεται το πλήθος των εφικτών σημείων *nf* και το πλήθος των μη εφικτών σημείων *ninf*.
4) Η συνάρτηση να υπολογίζει τα όρια των αξόνων X_1 και X_2 στις μεταβλητές *x1minmax* και *x2minmax*. Η τιμή *x1minmax(2)* να υπολογίζεται ως εξής. Αν το πρόβλημα δεν είναι εφικτό να είναι

$$x1\min\max(2) = \max\left\{\max\left\{Pinfeas(1,:)\right\},0,\frac{c(1)}{3}\right\}+1$$

Διαφορετικά να είναι

$$x1\min\max(2) = \max\left\{\max\left\{Pfeas(1,:)\right\},0,\frac{c(1)}{3}\right\}+1$$

5) Η τιμή *x1minmax(1)* να υπολογίζεται ως εξής. Αν το πρόβλημα δεν είναι εφικτό να είναι

$$x1\min\max(1) = \min\left\{\min\left\{Pinfeas(1,:)\right\},0,\frac{c(1)}{3}\right\}-0.5$$

Διαφορετικά να είναι

$$x1\min\max(1) = \min\left\{\min\left\{Pfeas(1,:)\right\},0,\frac{c(1)}{3}\right\}-0.5$$

Με ανάλογο τρόπο να υπολογίζονται τα στοιχεία *x2minmax(1)* και *x2minmax(2)*.

6) Οι κορυφές του πολυγώνου που θα χρωματιστεί με χρώμα κίτρινο να υπολογίζονται στη μεταβλητή εξόδου *polygono*. Να υπολογίζεται και το πλήθος, *npol*, των κορυφών του πολυγώνου. Προφανώς αν το πρόβλημα είναι εφικτό με περιορισμένη εφικτή περιοχή, τότε είναι *polygono=Pfeas*. Στην περίπτωση που η εφικτή περιοχή είναι απεριόριστη, στις κορυφές του πολυγώνου να περιλαμβάνονται δυο σημεία τα οποία είναι εφικτά, βρίσκονται πάνω στις ακτίνες της εφικτής περιοχής και στις ευθείες και X_1 = *x1minmax(2)* ή X_2 = *x2minmax(2)*. Επίσης να περιλαμβάνεται το σημείο *(x1minmax(2), x2minmax(2))*. Τα τρία τελευταία σημεία να καταγράφονται στη μεταβλητή εξόδου *Ponrays*. Να υπολογίζεται επίσης το πλήθος των στηλών της μήτρας *Ponrays*, στη μεταβλητή *nonrays*.

Λύση

Θα χρησιμοποιήσουμε πέντε συναρτήσεις που κατασκευάζουμε πρώτα. Η πρώτη ονομάζεται *intersection* η οποία υπολογίζει το σημείο τομής των ευθειών $a_1(1)\cdot x_1 + a_1(2)\cdot x_2 = b_1, a_2(1)\cdot x_1 + a_2(2)\cdot x_2 = b_2$ (αν οι ευθείες δεν τέμνονται σ' ένα σημείο η λύση X να είναι το κενό διάνυσμα X=[]). Ο κώδικας της είναι:

```
function [x,nofsolutions]=intersection(a1,b1,a2,b2)
% INTERSECTION : computes the solution of the system a1*x=b1 and
a2*x=b2
% SYNTAX  [x,nofsolutions]=intersection(a1,b1,a2,b2)
% Αν το σύστημα δεν έχει μοναδική λύση, είναι x=[].
%-----------------------------------------------------------
%------ Μεταβλητές εισόδου  ----------------------------
%-----------------------------------------------------------
% a1, a2 = 1x2 διανύσματα
% b1, bl = πραγματικοί αριθμοί
%-----------------------------------------------------------
%------  Μεταβλητές εξόδου  -----------------------------
%-----------------------------------------------------------
% x = η λύση του συστήματος
% nofsolutions = 0, αν το σύστημα δεν έχει λύση
%              = 1, αν το σύστημα έχει μοναδική λύση
%              = inf, αν το σύστημα έχει άπειρες λύσεις
%-------------------------------------------------------------------
% Γράφτηκε από Παπαρρίζο Κ. στις 23-5-1999.
% Τελευταία τροποποίηση στις 16-05-2000
%-------------------------------------------------------------------
if length(a1)~=2 | length(a2)~=2 | length(b1)~=1 | length(b2)~=1
```

```
   error('Αυτό που πληκτρολόγησες δεν είναι σύστημα 2x2')
end

% Υπολόγησε σύστημα a*x=b
a=[a1;a2];
b=[b1;b2];

% Αρχικοποιήσεις. Ταιριάζουν για την περίπτωση
% που το σύστημα είναι αδύνατο.
x=[];
nofsolutions=0;

if rank(a)==2
   % Το σύστημα έχει μια μοναδική λύση
   nofsolutions=1;
   x=[a\b]';
elseif (rank(a)==1)&(rank([a b])==1)
   % Το σύστημα έχει άπειρες λύσεις
   nofsolutions=inf;
end
```

Η δεύτερη ονομάζεται *isfeasible* η οποία ελέγχει αν το σημείο $x=(x_1,x_2)$ είναι εφικτό στο γραμμικό πρόβλημα με δεδομένα *A, b και eqin* (στη συνάρτηση αυτή χρησιμοποιείται η μεταβλητή εισόδου α, η οποία παίρνει τιμή 1 στην περίπτωση που το σημείο X είναι εφικτό και τιμή 0 στην περίπτωση που δεν είναι εφικτό). Ο κώδικας της συνάρτησης είναι:

```
function a=isfeasible(A,b,eqin,x)
% ISFEASIBLE : true if point x is feasible to the region defined by
A,b,eqin
% SYNTAX a=isfeasible(A,b,eqin,x)
%-------------------------------------------------------------
%------Input variables-------------------------------------
%-------------------------------------------------------------
% A = coefficient matrix
% b = RHS
% eqin(i)=0, 1, 2 (constraint i, is of type =, <=, >=, respectively)
% x = a point (row or column vector)
%-------------------------------------------------------------
%------Output variables------------------------------------
%-------------------------------------------------------------
% a = true if point x is feasible
%-------------------------------------------------------------------
% Written by Paparrizos K. on 23-5-99 Last  modified 10-11-2000
%-------------------------------------------------------------------

b=b(:);
eqin=eqin(:);
x=x(:);

[m n]=size(A);
%----------Testing  dimensions ---------------------------------
if length(b)~=m
   error('Length of b not equal to m')
end
```

```
if length(eqin)~=m
   error('Length of eqin not equal to m')
end
if length(x)~=n
   error('Length of x not equal to n')
end
%--------------------------------------------------------------------
a=1;
if any(x<0)
   a=0;
   return
end
equalities=find(eqin==0);
if (~isempty(equalities))&(any(A(equalities,:)*x~=b(equalities)))
   a=0;
   return
end
ltoet=find(eqin==1);
if (~isempty(ltoet))&(any(A(ltoet,:)*x>b(ltoet)))
   a=0;
   return
end
gtoet=find(eqin==2);
if (~isempty(gtoet))&(any(A(gtoet,:)*x<b(gtoet)))
   a=0;
   return
end
```

Η τρίτη ονομάζεται *uniquecol* η οποία δέχεται σαν μεταβλητή εισόδου μια μήτρα X και η οποία διαγράφει τις διπλές στήλες της μήτρας X. Ο κώδικας της *uniquecol* είναι:

```
function [x,double] = uniquecol(x)
% UNIQUECOL: deletes duplicated columns of the matrix x.
% SYNTAX [x,double] = uniquecol(x)
% Two columns are equal if the norm of their difference is small
%-----------------------------------------------------------
%------Input variables-------------------------------------
%-----------------------------------------------------------
% x = a matrix
%-----------------------------------------------------------
%------Output variables------------------------------------
%-----------------------------------------------------------
% x the unique columns of the input matrix x
% double = a row vector containing the indeces of the deleted columns
%-----------------------------------------------------------------
% Written by Paparrizos K. on 13-5-99 Last  modified 16-05-2000
%-----------------------------------------------------------------


[m,n]=size(x);
double = [];
for k =1:n-1
   xx=x(:,k);
   for l=k+1:n
      if norm(xx-x(:,l))< 10*eps
         double = [double l];
```

```
      end
   end
end
if ~isempty(double)
   x(:,double) = [];
end
```

Η τέταρτη ονομάζεται *basfeasinfeasR2* η οποία δέχεται σαν δεδομένα εισόδου τα στοιχεία *A, b, eqin* και υπολογίζει όλα τα σημεία τα οποία είναι τομές δύο ευθειών τα οποία χωρίζει σε εφικτά και μη εφικτά.(εφικτά στήλες της μήτρας *Pfeas*, μη εφικτά στήλες της μήτρας *Pinfeas*). Ο κώδικας της *basfeasinfeasR2* είναι.

```
function [Pfeas,Pinfeas]=basfeasinfeasR2(A,b,eqin)
[m,n]=size(A);
b=b(:);eqin=eqin(:);

%Πρόσθεσε γραμμές στα A,b
A1=[0 1;1 0;A];
b1=[0;0;b];
eqin1=[2;2;eqin];

Pfeas=[];Pinfeas=[];
%Υπολόγισε τις συντεταγμένες όλων των βασικών σημείων
for i=1:m+1
   for j=(i+1):m+2
      B=[A1(i,:);A1(j,:)];
      if det(B)==0
         break
      end
      bb=[b1(i);b1(j)];
      x=inv(B)*bb;
      A2=A1;b2=b1;eqin2=eqin1;
      A2([i j],:)=[];b2([i j])=[];eqin2([i j])=[];
      a=isfeasible(A2,b2,eqin2,x);
      if a==1
         Pfeas=[Pfeas x];
      else
         Pinfeas=[Pinfeas x];
      end
   end
end

% Σβήσε διπλές στήλες
Pfeas = uniquecol(Pfeas);
Pinfeas = uniquecol(Pinfeas);
```

Η πέμπτη συνάρτηση που κατασκευάζουμε είναι η *raypoints* η οποία δέχεται σαν μεταβλητές εισόδου τα δεδομένα των περιορισμών ενός γραμμικού προβλήματος, δηλαδή τα στοιχεία *c, A, b, eqin* και δύο αριθμούς X και Y έτσι ώστε ο X να είναι μεγαλύτερος από τη μέγιστη τετμημένη και ο Y μεγαλύτερος από τη μέγιστη

τεταγμένη των εφικτών βασικών σημείων του προβλήματος. Η συνάρτηση αυτή υπολογίζει τα σημεία τομής (α, β) των ακτίνων με τις ευθείες $X_1 = X$ και $X_2 = Y$. Τα εφικτά σημεία που ικανοποιούν τη σχέση $a \le X$ και $\beta \le Y$ να τα καταγράφει σαν στήλες της μήτρας *Ponrays*. Στην ίδια μήτρα σαν Τρίτη στήλη να καταγράφει και το σημείο (X, Y) στην περίπτωση που είναι εφικτό.

Ο κώδικας της *raypoints* δίνεται παρακάτω.

```
function [Ponrays]=raypoints(A,b,eqin,x,y)
[m,n]=size(A);
Ponrays=[];
for i=1:m
   [z,numofsolutions]=...
      intersection(A(i,:),b(i),[1 0],x);
   if numofsolutions==1 & isfeasible(A,b,eqin,z)...
         & z(2)<= y
      Ponrays=[Ponrays z'];
   end
end
z=[x 0];
if isfeasible(A,b,eqin,z)
   Ponrays=[Ponrays z'];
end
for i=1:m
   [z,nofsolutions]=...
      intersection(A(i,:),b(i),[0 1],y);
   if numofsolutions==1 & isfeasible(A,b,eqin,z)...
         & z(1)<= x
      Ponrays=[Ponrays z'];
   end
end
z=[0 y];
if isfeasible(A,b,eqin,z)
   Ponrays=[Ponrays z'];
end
z=[x y];
if isfeasible(A,b,eqin,z)
   Ponrays=[Ponrays z'];
end
[Ponrays,double] = uniquecol(Ponrays);
```

Στη συνέχεια κατασκευάζουμε τη συνάρτηση *solverLPinR2*.

Η σειρά με την οποία καλούνται οι συναρτήσεις *basfeasinfeasR2* και *raypoints* (μέσα στην *solverLPinR2*) φαίνεται καθαρά στον επόμενο κώδικα.

```
function [adarap,zopt,xopt,Pfeas,Pinfeas,Ponrays,polygono,...
          nf,ninf,nonrays,npol,x1minmax,x2minmax]...
          =solveLPinR2(minmax,c,A,b,eqin)
% Χώρισε εφικτά και μη εφικτά βασικά σημεία
[Pfeas,Pinfeas]=basfeasinfeasR2(A,b,eqin);
[mf,nf]=size(Pfeas);
[minf,ninf]=size(Pinfeas);
%Υπολόγισε όρια αξόνων
```

```
if nf==0
   x1minmax(1)=min([Pinfeas(1,:) c(1)/3 0])-0.5;
   x1minmax(2)=max([Pinfeas(1,:) c(1)/3 0])+.5;
   x2minmax(1)=min([Pinfeas(2,:) c(2)/3 0])-0.5;
   x2minmax(2)=max([Pinfeas(2,:) c(2)/3 0])+.5;
else
   x1minmax(1)=min([Pfeas(1,:) c(1)/3 0])-0.5;
   x1minmax(2)=max([Pfeas(1,:)c(1)/3 0])+1;
   x2minmax(1)=min([Pfeas(2,:) c(2)/3 0])-0.5;
   x2minmax(2)=max([Pfeas(2,:)c(2)/3 0])+1;
end
% Βρες αν η εφικτή περιοχή είναι απεριόριστη
[Ponrays]=raypoints(A,b,eqin,x1minmax(2),x2minmax(2));
[monrays,nonrays]=size(Ponrays);
% Προσδιόρισε αν το πρόβλημα είναι αδύνατο,
% βέλτιστο ή απεριόριστο
adarap=-1;
if nf>0,adarap=2;end
c1=c;
if minmax==1,c1=-c;end
zvfeas=[];zvonrays=[];
if nf>0
   for i=1:nf
      zvfeas(i)=c1*Pfeas(:,i);
   end
end
if nonrays>0
   for i=1:nonrays
      zvonrays=c1*Ponrays(:,i);
   end
end
zopt=0;xopt=[];
if adarap~=-1
   [zopt,index]=min(zvfeas);
   adarap=0;
   xopt=Pfeas(:,index);
   if nonrays>0
      z1=min(zvonrays);
      if z1<zopt
         adarap=1;
      end
   end
end
if minmax==1,zopt=-zopt;end
% Προσδιόρισε το πολυγονο που θα γεμίσει με χρώμα
polygono=[];
if nf>0,polygono=Pfeas;end
if nonrays>0,polygono=[Pfeas Ponrays];end
[mpol,npol]=size(polygono);
```

Οι μεταβλητές εξόδου της συνάρτησης θα χρησιμοποιηθούν σε συναρτήσεις γραφικών σε συναρτήσεις γραφικών που θα αναπτυχθούν στην επόμενη ενότητα. Εδώ θα ελέγξουμε τη συνάρτηση χρησιμοποιώντας μόνο τις τρεις μεταβλητές εξόδου. Λύνουμε το παρακάτω πρόβλημα με τη συνάρτηση *solverLPinR2* και έχουμε.

```
>> minmax=1;
>> c=[1 3];
>> A=[1 1;2 -1;1 -1;2 1];
>> b=[5 -2 1 4];
>> eqin=[1 2 1 2];
>> [adarap,zopt,xopt]=solveLPinR2(minmax,c,A,b,eqin)

adarap =

     0


zopt =

   13.0000


xopt =

    1.0000
    4.0000
```

Θέτοντας

```
>> eqin(1)=2;
```

Το πρόβλημα γίνεται απεριόριστο

```
>> [adarap,zopt,xopt]=solveLPinR2(minmax,c,A,b,eqin)

adarap =

     1


zopt =

   13.0000


xopt =
```

```
   1.0000
   4.0000
```

Και θέτοντας

```
>> eqin(4)=1;
```

Το πρόβλημα γίνεται αδύνατο

```
>> [adarap,zopt,xopt]=solveLPinR2(minmax,c,A,b,eqin)

adarap =

   -1


zopt =

    0


xopt =

    []
```

Στη συνέχεια θα αναπτύξουμε τη συνάρτηση της γραφικής επίλυσης του γραμμικού προβλήματος.

Πριν περιγράψουμε τη συνάρτηση της γραφικής επίλυσης του γραμμικού προβλήματος θα αναπτύξουμε μια βοηθητική συνάρτηση η οποία ζωγραφίζει μερικές αντικειμενικές ευθείες. Οι λεπτομέρειες της συνάρτησης αυτής περιγράφονται στο επόμενο παράδειγμα.

Παράδειγμα 5.2.2

Να γραφεί συνάρτηση με όνομα *drawobjlines*, η οποία να κάνει τα εξής. Να δέχεται σαν μεταβλητές εισόδου τα δεδομένα *adarap, c, zopt, xopt* και ένα σύνολο σημείων σαν στήλες της μήτρας *polygono*. Επίσης μεταβλητές εισόδου να είναι τα όρια των αξόνων X_1 και X_2 αποθηκευμένες στα διαστήματα διανύσματα $X_1minmax$ και $X_2minmax$.

Η συνάρτηση και ζωγραφίζει το διάνυσμα c/3 (ζωγραφίζεται το 1/3 του διανύσματος c για λόγους αισθητικής) και την αντικειμενική γραμμή που περνάει από την αρχή των αξόνων. Αν το πρόβλημα είναι εφικτό να ζωγραφίζει τις αντικειμενικές ευθείες που διέρχονται από τα σημεία που είναι στήλες της μήτρας *polygono*. Οι αντικειμενικές ευθείες να ζωγραφίζονται με διακεκομμένες κόκκινες γραμμές.

Αν το πρόβλημα είναι βέλτιστο να ζωγραφίζεται η ευθεία *c*xopt = zopt* με συνεχή κόκκινη γραμμή και να τοποθετείται ένα μαύρο τετράγωνο στο βέλτιστο σημείο *xopt*.

Λύση

Ο παρακάτω κώδικας της συνάρτησης είναι εμπλουτισμένος με τα κατάλληλα σχόλια ώστε η κατανόηση του να είναι εύκολη.

```
exampleLP1;
function drawobjlines(adarap,c,zopt,xopt,x1minmax,x2minmax,polygono)
hold on
%   Ζωγράφισε (1/3)c
compass(c(1)/3,c(2)/3,'-k')
text(c(1)/3+0.05,c(2)/3,'c/3','FontWeight','bold')
% Ζωγράφισε αντικειμενική z=0
[x1,x2]=endpoints(c,0,x1minmax,x2minmax);
plot(x1,x2,'--r')
if adarap~=-1
   %Ζωγράφισε αντικειμενικές γραμμές
   for i=1:length(polygono)
      z=c*polygono(:,i);
      [x1,x2]=endpoints(c,z,x1minmax,x2minmax);
      plot(x1,x2,'--r')
   end
end
if adarap==0
   % Ζωγράφισε τη βέλτιστη αντικειμενική συνάρτηση
   [x1,x2]=endpoints(c,zopt,x1minmax,x2minmax);
   plot(x1,x2,'-r')
   plot(xopt(1),xopt(2),'sk','MarkerSize',7,'MarkerFaceColor','k')
end
hold off
```

Δίνουμε επίσης και τη συνάρτηση *endoints* η οποία υπολογίζει τις συντεταγμένες των άκρων ενός ευθυγράμμου τμήματος AB (το τμήμα AB είναι η τομή μιας ευθείας $a(1)\cdot x_1 + a(2)\cdot x_2 = b$ και ενός ορθογωνίου παραλληλογράμμου). Ο κώδικας της συνάρτησης αυτής είναι:

```
function [x1,x2]=endpoints(a,b,x1minmax,x2minmax)
% ENDPOINTS : Υπολογίζει σημεία τομής ευθείας και περιμέτρου
ορθογωνίου.
% Αν δεν υπάρχουν σημεία τομής, θέτει x1=[] και x2=[].
% Καλεί τη συνάρτηση intersection
% Written by K. Paparrizos  on October 3, 2000.
```

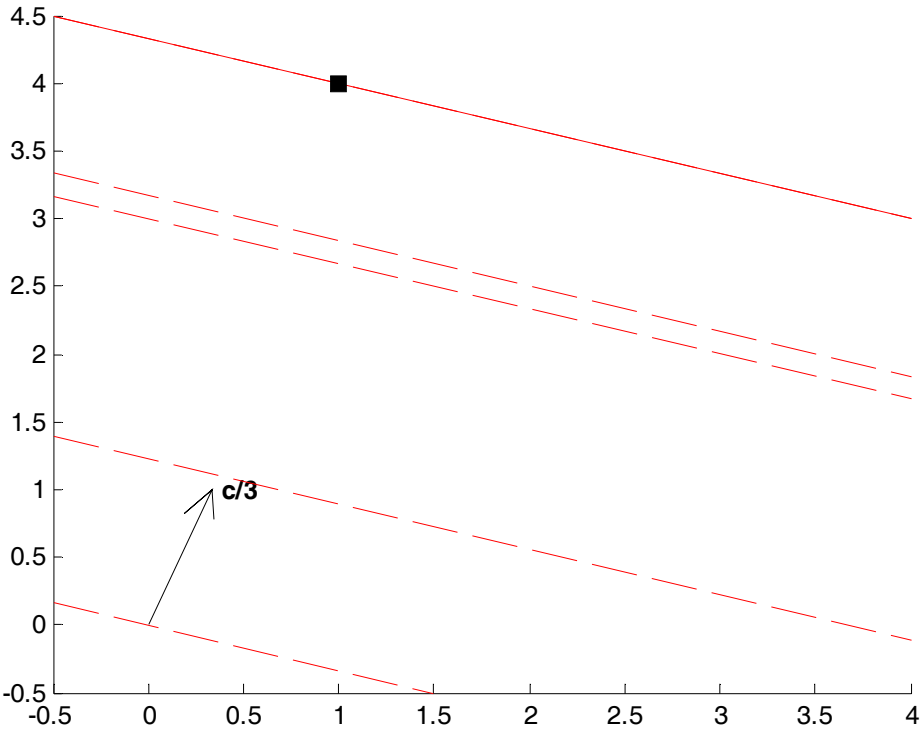
```
if (a(1)==0)&(a(2)==0)
   error('Λάθος δεδομένα. α(1)=α(2)=0')
end
x1=[]; x2=[];

% Ευθεία κάθετη στον άχονα x1
if a(2)~=0 & a(1)==0
   ba2=b/a(2);
   if (ba2<x2minmax(1)) | (ba2>x2minmax(2))
      return
   else
    x1=x1minmax;
    x2=[ba2 ba2];
    return
    end
end

% Ευθεία κάθετη στον άχονα x2
if a(1)~=0 & a(2)==0
   ba1=b/a(1);
   if(ba1<x1minmax(1)) | (ba1>x1minmax(2))
      return
    else
    x1=[ba1 ba1];
    x2=x2minmax;
    return
   end
end

% Η ευθεία τέμνει τους δυο άχονες
% Υπολόγισε τα σημεία a1,a2 τομής της ευθείας
% με τις κάθετες πλευρές του ορθογωνίου
[a1]=intersection(a,b,[1 0],x1minmax(1));
[a2]=intersection(a,b,[1 0],x1minmax(2));
% Υπολόγισε τα σημεία a3,a4 τομής της ευθείας
% με τις οριζόντιες πλευρές του ορθογωνίου
[a3]=intersection(a,b,[0 1],x2minmax(1));
[a4]=intersection(a,b,[0 1],x2minmax(2));
% Βάλε τα σημεία σαν στήλες σε μια μήτρα
A=[a1' a2' a3' a4'];
% Διάγραψε τα σημεία (σήλες) που δέν ανήκουν στο ορθογώνιο
%Χρησιμοποίησε βοηθητικό διάνυσμα t. t(i)==0 σημαίνει ότι
% το σημείο ai δεν ανήκει στο ορθογώνιο
t=[0 0 0 0];% Προσδιόρισε πρώτα το t
for i=1:4
   if x1minmax(1)<=A(1,i) & A(1,i)<=x1minmax(2)...
         & x2minmax(1)<=A(2,i)&A(2,i)<=x2minmax(2)
      t(i)=1;
   end
end

% Τώρα διάγραψε τα σημεία εκτός ορθογωνίου
tt=find(t==0);
A(:,tt)=[];
% Αν η ευθεία και το ορθογώνιο τέμνονται,
% γράψε  τα σημεία τομής στα διανύσματα x1,x2.
if ~isempty(A)
   [mm,nn]=size(A);
   if nn==1
```

```
        A=[A A];
    end
    x1=A(1,[1 2]);
    x2=A(2,[1 2]);
end
```

.

Χρησιμοποιώντας σαν *polygono* τη μήτρα *Pfeas* για το πρόβλημα του αρχείου *exampleLP1* η συνάρτηση παράγει την παράσταση της Εικόνας 5.1. Η εικόνα παράγεται με τις επόμενες εντολές.

```
>> minmax=1;
>> c=[1 3];
>> A=[1 1;2 -1;1 -1;2 1];
>> b=[5 -2 1 4];
>> eqin=[1 2 1 2];
>> [adarap,zopt,xopt,Pfeas,Pinfeas,Ponrays,polygono,...
      nf,ninf,nonrays,npol,x1minmax,x2minmax]...
      =solveLPinR2(minmax,c,A,b,eqin);
>> clf
>> drawobjlines(adarap,c,zopt,xopt,x1minmax,x2minmax,polygono)
```



Εικόνα 5.1

Είμαστε τώρα σε θέση να περιγράψουμε μια ολοκληρωμένη συνάρτηση για τη γραφική επίλυση του γραμμικού προβλήματος.

Παράδειγμα 5.2.3

Να γραφεί συνάρτηση με όνομα *drawLP,* η οποία να επιλύει γραφικά κάθε γραμμικό πρόβλημα στο επίπεδο. Πιο συγκεκριμένα, η συνάρτηση να προσδιορίζει την εφικτή περιοχή και να τη γεμίζει με κίτρινο χρώμα. Στην περίπτωση που η εφικτή περιοχή είναι απεριόριστη να περιλαμβάνει δυο σημεία πάνω στις ακτίνες και το σημείο (*X1minmax(2), X2minmax(2)*) στην περίπτωση που είναι εφικτό. Τα όρια των αξόνων *X1minmax* και X2minmax να προσδιορίζονται έτσι ώστε να περιλαμβάνουν κάθε εφικτό σημείο την αρχή των αξόνων και το διάνυσμα c/3. Στην περίπτωση που το πρόβλημα είναι αδύνατο τα όρια των αξόμων να συμπεριλαμβάνουν κάθε βασικό σημείο. Να ζωγραφίζει τους άξονες X_1 και X_2 τοποθετώντας βέλη και γράφοντας X_1 και X_2 δίπλα στα βέλη. Οι περιορισμοί του προβλήματος να ζωγραφίζονται με τη συνάρτηση *drawing*. Γι' αυτό το λόγο σαν μεταβλητές εισόδου, εκτός από τα δεδομένα *minmax, c, A, b, equin*, να συμπεριλαμβάνονται και οι μεταβλητές *step* και *alength.* Επίσης να ζωγραφίζεται το διάνυσμα c/3 και να γράφονται αντικειμενικές γραμμές με τη συνάρτηση *drawobjlines.*

Λύση

Αρχικά θα φτιάξουμε δύο συναρτήσεις την *drawannotaxis* η οποία ζωγραφίζει τους άξονες X_1 και X_2 και η οποία έχει κώδικα:

```
function drawannotaxis(x1minmax,x2minmax)
plot(x1minmax,[0 0],'-k','LineWidth', 1.2)
hold on
plot([0 0],x2minmax,'-k','LineWidth', 1.2)
text(x1minmax(2), 0,'>','FontWeight','bold',...
                        'HorizontalAlignment','right')
text(0, x2minmax(2),'>','FontWeight','bold',...
                        'HorizontalAlignment','right',...
                        'Rotation',90)
text(x1minmax(2),0,'\chi_{1 }','FontWeight','bold',...
                            'HorizontalAlignment','right',...
                            'VerticalAlignment','top')
text(0,x2minmax(2),'\chi_{2 }','FontWeight','bold',...
                            'HorizontalAlignment','right',...
                            'VerticalAlignment','top')
hold off
```

Και την *drawineq* η οποία επιλύει ανισότητες της μορφής $a(1)\cdot x_1 + a(2)\cdot x_2 \geq b$ ή της μορφής $a(1)\cdot x_1 + a(2)\cdot x_2 \leq b_2$ ζωγραφίζοντας την ευθεία $a(1)\cdot x_1 + a(2)\cdot x_2 = b$ και η οποία δείχνει το σωστό ημιεπίπεδο ζωγραφίζοντας βέλη κάθετα στην ευθεία. (μεταβλητές εισόδου είναι το διάνυσμα α, ο αριθμός b, ο τύπος της ανισότητας στη μεταβλητή type, το μήκος των τόξων alength, η απόσταση μεταξύ των τόξων, step, και τα όρια των αξόνων x1minmax και x2minmax). Η συνάρτηση αυτή συνοδεύεται από τη συνάρτηση *interpoints* η οποία δέχεται σαν μεταβλητές εισόδου τις τετμημένες x_1 και τις τεταγμένες x_2, δύο σημείων Α και Β και ένα θετικό αριθμό *numofpoints* και τοποθετεί στο ευθύγραμμο τμήμα ΑΒ *numofpoints* ισαπέχοντα σημεία. Ο κώδικας της συνάρτησης *interpoints* είναι:

```
function [x1,x2]=interpoints(x,y,numofpoints)

if x(1)==x(2)
   x1=x(1)*ones(1,numofpoints);
   x2=linspace(y(1),y(2),numofpoints);
else
   x1=linspace(x(1),x(2),numofpoints);
   if y(1)==y(2)
      x2=y(1)*ones(1,numofpoints);
   else
      x2=linspace(y(1),y(2),numofpoints);
   end
end
```

και ο κώδικας της συνάρτησης *drawineq* είναι:

```
function drawineq(a,b,type,x1minmax,x2minmax,step,alength)
% DRAWINEQ : draws an inequality constraint
% SYNTAX drawconstrADV(a,b,type,x1minmax,x2minmax,step,alength)
% It draws the constraint a*x = a(1)*x1 + a(2)*x2 [type] b, where
% type = 1 means <= and type = 2 means >=. Arrows of length alength
% vertical to the line are drawn.
% The distance between arrows is equal to step
%-------------------------------------------------------------
%------Input variables----------------------------------------
%-------------------------------------------------------------
% a = a 1x2 vector
% b = a real number
% type = 1 means inequality of type <= and =2 means inequality of
type >=
% x1minmax = a 1x2 vector with the lower and upper bound of axis x1
% x2minmax = a 1x2 vector with the lower and upper bound of axis x2
% step = distance between arrows
% alength = length of arrows
%-------------------------------------------------------------------
% Written by Paparrizos K. on 23-5-99. Last  modified 16-05-2000,3-1-
2001
%-------------------------------------------------------------------
```

```
hold on
%   Compute endpoints of the line
[x1,x2]=endpoints(a,b,x1minmax,x2minmax);

%   Draw the corresponding line
plot(x1,x2,'k-')

%   Compute number of arrows
A=[x1(1) x2(1)];B=[x1(2) x2(2)];
numofarrows=round(norm(A-B,2)/step)+1;

%   Compute tails of arrows
[x,y]=interpoints(x1,x2,numofarrows);

%   Compute direction and length of a single arrow
if type==2
   arrow=a/norm(a,2);
elseif type==1
   arrow=-a/norm(a,2);
else
   error('This is not an inequality constraint')
end
arrow=alength*arrow;

%   Compute all arrows
Dx1=zeros(numofarrows,1);
Dx2=zeros(numofarrows,1);
for i=1:numofarrows
   Dx1(i)=arrow(1);
   Dx2(i)=arrow(2);
end

%   Draw the arrows
quiver(x',y',Dx1,Dx2,0,'b-')

axis('equal')
hold off
```

Οι λειτουργίες της συνάρτησης *drawLP* γίνονται με τη σειρά που περιγράφεται στον παρακάτω κώδικα. Ο κώδικας είναι εμπλουτισμένος με κατάλληλα σχόλια για την εύκολη κατανόηση του.

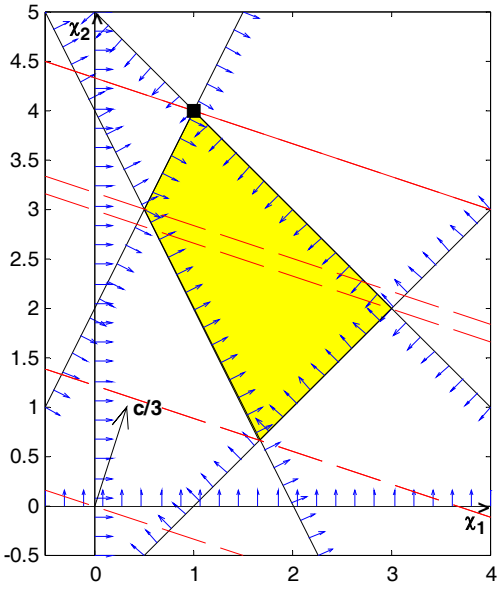
```
function drawLP(minmax,c,A,b,eqin,step,alength)

%DRAWLP: solves grafically in R2 an lp in canonical form
c=c(:)';b=b(:);eqin=eqin(:);
[m,n]=size(A);
if n>2,error('n greater than 2'),end

% Λύσε το πρόβλημα χωρίς τη χρήση αλγορίθμων
[adarap,zopt,xopt,Pfeas,Pinfeas,Ponrays,polygono,...
          nf,ninf,nonrays,npol,x1minmax,x2minmax]...
          =solveLPinR2(minmax,c,A,b,eqin);
```

```
% Βάλε με σειρά τις κορυφές του πολυγώνου
% και γέμισέ το με χρώμα
clf,hold on
if npol>1
   K=convhull(polygono(1,:),polygono(2,:));
   polygono=polygono(:,K);
   fill(polygono(1,:),polygono(2,:),'y')
end

% Ζωγράφισε άξονες χ1,χ2
drawannotaxis(x1minmax,x2minmax)

% Ζωγράφισε τις ανισότητες
A1=[A;[1 0];[0 1]];,b1=[b;0;0];,eqin1=[eqin;2;2];
for i=1:m+2
   drawineq(A1(i,:),b1(i),eqin1(i),x1minmax,x2minmax,step,alength)
end

% Ζωγράφισε c/3 και αντικειμενικές γραμμές
drawobjlines(adarap,c,zopt,xopt,x1minmax,x2minmax,polygono)


% Χρησιμοποίησε τα όρια αξόνων
axis equal,box on
axis([x1minmax(1) x1minmax(2) x2minmax(1) x2minmax(2)])
hold off
```

Οι παρακάτω εντολές εφαρμόζουν τη συνάρτηση στο γραμμικό πρόβλημα.

```
>> minmax=1;
>> c=[1 3];
>> A=[1 1;2 -1;1 -1;2 1];
>> b=[5 -2 1 4];
>> eqin=[1 2 1 2];
>> clf
>> drawLP(minmax,c,A,b,eqin,0.2,0.175)
```

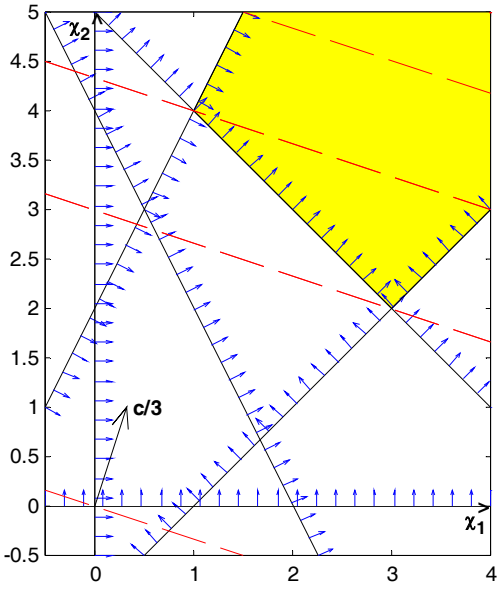
Το αποτέλεσμα τω εντολών φαίνεται στην Εικόνα 5.2.



Εικόνα 5.2

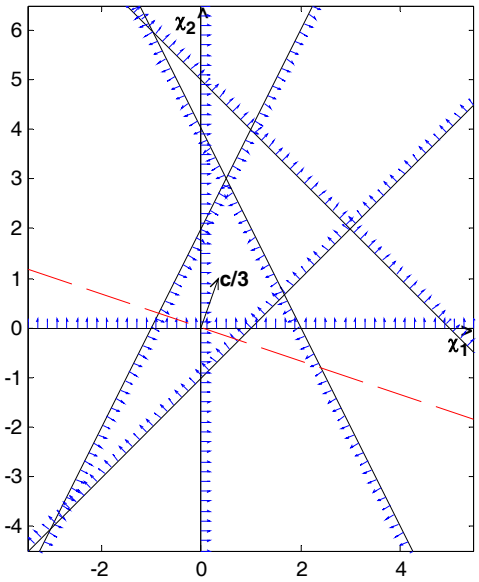
Θα επιβεβαιώσουμε την ορθότητα της συνάρτησης *drawLP* εφαρμόζοντας την σε δυο ακόμη γραφικά προβλήματα, ένα απεριόριστο (Εικόνα 5.3) και ένα αδύνατο (Εικόνα 5.4). Οι επόμενες εντολές υλοποιούν τον παρακάτω στόχο.

```
>> minmax=1;
>> c=[1 3];
>> A=[1 1;2 -1;1 -1;2 1];
>> b=[5 -2 1 4];
>> eqin(1)=2; %Πρόβλημα απεριόριστο
>> drawLP(minmax,c,A,b,eqin,0.2,0.175)
```



Εικόνα 5.3

```
>> clf
>> eqin(4)=1; %Πρόβλημα αδύνατο
>> drawLP(minmax,c,A,b,eqin,0.2,0.175)
```



Εικόνα 5.4

Η συνάρτηση *drawLP* είναι ικανοποιητική. Μπορούμε όμως να προσθέσουμε τίτλους και χαρακτηριστικά στα βασικά εφικτά και μη εφικτά σημεία ώστε να βελτιωθεί η

εμφάνιση. Το αρχείο εντολών *annotregion* γράφει στο κέντρο της εφικτής περιοχής το κείμενο «FEASIBLE AREA». Ο κώδικας του είναι ο παρακάτω.

```
% Αν η εφικτή περιοχή είναι κενή, επέστρεψε.
if adarap==-1
   disp(   'The problem is not feasible')
   axis equal
   axis([x1minmax(1) x1minmax(2) x2minmax(1) x2minmax(2)])
   box on
   hold off
   return
end
% Γράψε κείμενο
xmean=mean(polygono(1,1:length(polygono)-1));
ymean=mean(polygono(2,1:length(polygono)-1));
text(xmean,ymean+.15,'FEASIBLE','FontWeight','bold',...
                        'HorizontalAlignment','center')
text(xmean,ymean-.15,'AREA','FontWeight','bold',...
                        'HorizontalAlignment','center')
```

Το αρχείο εντολών *annotfeasinfeas* γράφει ένα κύκλο σε κάθε μη εφικτό σημείο και τοποθετεί μια μεγάλη μαύρη τελεία σε κάθε βασικό εφικτό σημείο. Ο πλήρης κώδικας του είναι ο παρακάτω.

```
hold on
% Σημείωσε τα μη εφικτά σημεία
if ninf>0
   for i=1:ninf
      plot(Pinfeas(1,i),Pinfeas(2,i),'ok','MarkerSize',5)
   end
end
% Σημείωσε τα εφικτά σημεία
if nf>0
   for i=1:nf
      plot(Pfeas(1,i),Pfeas(2,i),'.k','MarkerSize',15)
   end
end
hold off
```

Όταν οι δυο συναρτήσεις *annotfeasinfeas* και *annotregion* εισαχθούν σε κατάλληλες θέσεις μέσα στη συνάρτηση *drawLP*, προκύπτει η συνάρτηση *drawLPannot*, της οποίας ο πλήρης κώδικας είναι ο παρακάτω.

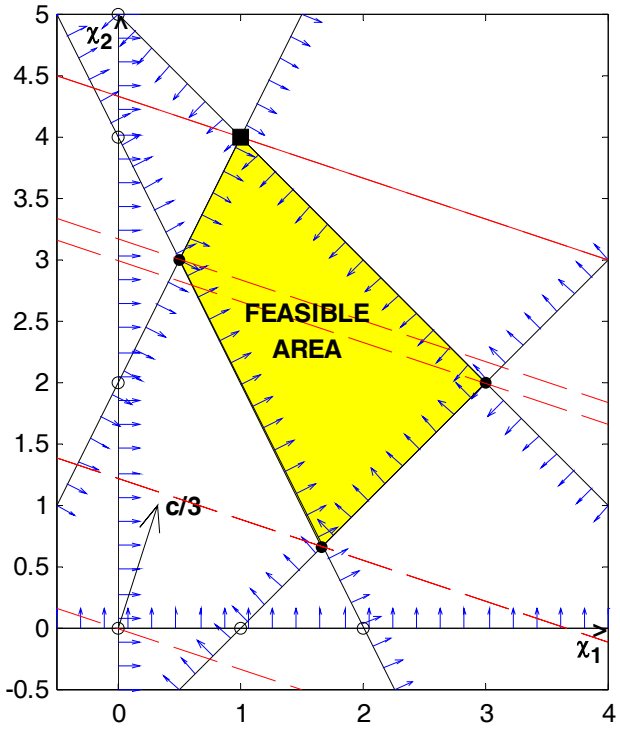
```
function drawLPannot(minmax,c,A,b,eqin,step,alength)

%DRAWLP: solves grafically in R2 an lp in canonical form
```

```
c=c(:)';b=b(:);eqin=eqin(:);
[m,n]=size(A);
if n>2,error('n greater than 2'),end

% Λύσε το πρόβλημα χωρίς τη χρήση αλγορίθμων
[adarap,zopt,xopt,Pfeas,Pinfeas,Ponrays,polygono,...
          nf,ninf,nonrays,npol,x1minmax,x2minmax]...
          =solveLPinR2(minmax,c,A,b,eqin);

% Βάλε με σειρά τις κορυφές του πολυγώνου
% και γέμισέ το με χρώμα
clf,hold on
if npol>1
   K=convhull(polygono(1,:),polygono(2,:));
   polygono=polygono(:,K);
   fill(polygono(1,:),polygono(2,:),'y')
end

% Ζωγράφισε άξονες χ1,χ2
drawannotaxis(x1minmax,x2minmax)

% Ζωγράφισε τις ανισότητες
A1=[A;[1 0];[0 1]];,b1=[b;0;0];,eqin1=[eqin;2;2];
for i=1:m+2
   drawineq(A1(i,:),b1(i),eqin1(i),x1minmax,x2minmax,step,alength)
end

% Χώρισε εφικτά και μη εφικτά σημεία
annotfeasinfeas

% Ζωγράφισε c/3 και αντικειμενικές γραμμές
drawobjlines(adarap,c,zopt,xopt,x1minmax,x2minmax,polygono)

%Γράψε μέσα στην εφικτή πειοχή 'FEASIBLE AREA'
annotregion

% Χρησιμοποίησε τα όρια αξόνων
axis equal,box on
axis([x1minmax(1) x1minmax(2) x2minmax(1) x2minmax(2)])
hold off
```

Η εφαρμογή της συνάρτησης *drawLPannot* γίνεται με τις εντολές.

```
>> minmax=1;
>> c=[1 3];
>> A=[1 1;2 -1;1 -1;2 1];
>> b=[5 -2 1 4];
>> eqin=[1 2 1 2];
>> clf
>> drawLPannot(minmax,c,A,b,eqin,0.2,0.175)
```

Το αποτέλεσμα των εντολών φαίνεται στην εικόνα 5.3.



Εικόνα 5.3

ΚΕΦΑΛΑΙΟ 6
ΠΕΡΙΓΡΑΦΗ ΚΑΙ ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΟΥ SIMULINK (ΕΚΔΟΣΕΙΣ 6.Χ ΚΑΙ ΑΝΩ)

6.1. Γενικά

Το SIMULINK είναι ένα λογισμικό πακέτο που επιτρέπει τη μοντελοποίηση, προσομοίωση και ανάλυση δυναμικών συστημάτων. Υποστηρίζει γραμμικά και μη γραμμικά συστήματα, μοντελοποιημένα σε συνεχή ή διακριτό χρόνο, ή ακόμη και υβριδικά συστήματα (εν μέρει μοντελοποιημένα σε συνεχή και εν μέρει σε διακριτό χρόνο). Υποστηρίζονται ακόμη συστήματα με τμηματικά διαφορετικούς χρόνους δειγματοληψίας.

Για τη μοντελοποίηση, το SIMULINK παρέχει ένα γραφικό περιβάλλον διεπαφής (GUI) που επιτρέπει την κατασκευή μοντέλων ως δομικών διαγραμμάτων, χρησιμοποιώντας λειτουργίες click-and-drag του ποντικιού. Το SIMULINK περιλαμβάνει ένα πλήθος βιβλιοθηκών δομικών στοιχείων (blocks), τα βασικότερα από τα οποία είναι οι πηγές (sources), τα στοιχεία «απορρόφησης» (sinks), τα συνεχή γραμμικά στοιχεία, τα μη γραμμικά στοιχεία και τα στοιχεία σημάτων και συστημάτων. Είναι επίσης δυνατή η τροποποίηση και η δημιουργία νέων δομικών στοιχείων από το χρήστη.

Τα μοντέλα SIMULINK είναι *ιεραρχικά* (ένα μοντέλο μπορεί να περιέχει μπλοκ τα οποία περιέχουν με τη σειρά τους άλλα μπλοκ), έτσι μπορούν να ιδωθούν σε διάφορα επίπεδα. Ένα σύστημα που έχει ιεραρχική δομή μπορεί να ιδωθεί αρχικά σε υψηλό επίπεδο ως ένα σύνολο διασυνδεμένων υποσυστημάτων, κάθε ένα από τα οποία μοντελοποιείται ως ένα μπλοκ. Στη συνέχεια, κάνοντας διπλό κλικ με το ποντίκι στα επί μέρους μπλοκ, ο χρήστης μπορεί να κατέβει σε χαμηλότερα επίπεδα ώστε να δει αυξανόμενους βαθμούς λεπτομέρειας.

Μετά τη δημιουργία ενός μοντέλου, είναι δυνατή η προσομοίωση του, χρησιμοποιώντας μια από τις διάφορες μεθόδους ολοκλήρωσης που παρέχει το SIMULINK. Χρησιμοποιώντας παλμογράφους (scopes) και άλλα μπλοκ απεικόνισης,

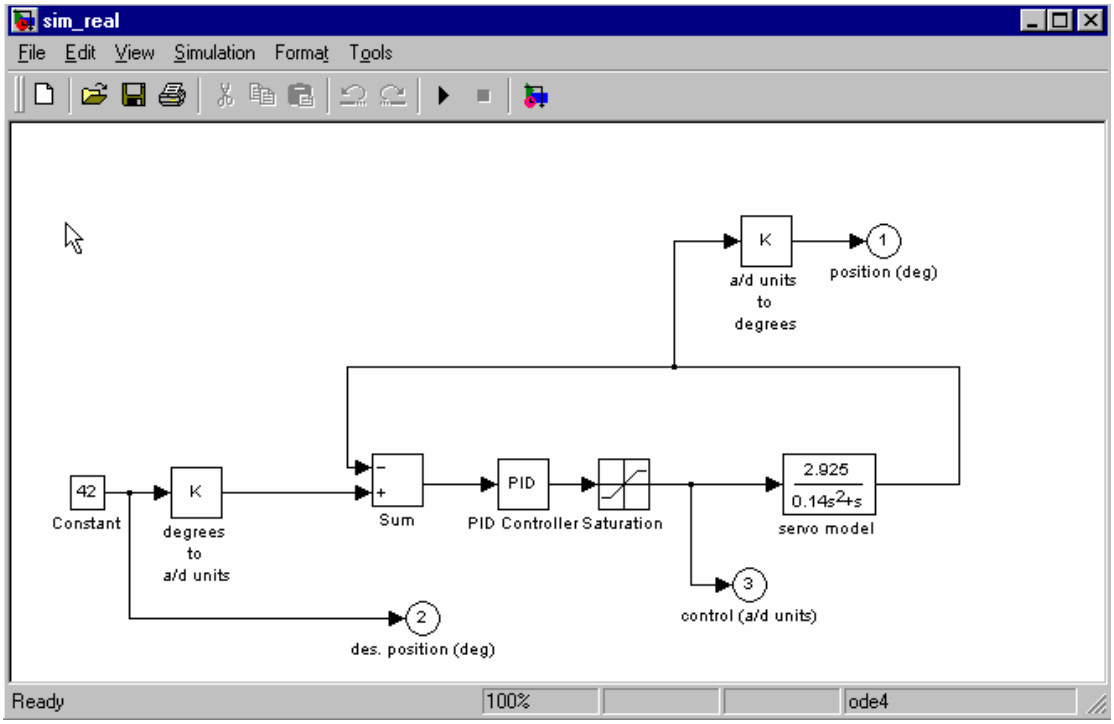
είναι δυνατή η παρακολούθηση των αποτελεσμάτων της προσομοίωσης καθώς αυτή εξελίσσεται. Επιπλέον, είναι δυνατή η εξαγωγή αποτελεσμάτων της προσομοίωσης στο χώρο εργασίας της MATLAB για περαιτέρω επεξεργασία. Είναι ακόμη δυνατή η χρήση του SIMULINK για προσομοίωση αλλά και έλεγχο συστημάτων σε πραγματικό χρόνο, μέσω της εργαλειοθήκης πραγματικού χρόνου (Real Time Workshop).

Στη συνέχεια θα δοθούν κάποιες βασικές οδηγίες χρήσης με έμφαση στην προσομοίωση έτοιμων μοντέλων SIMULINK και θα γίνει μια συνοπτική περιγραφή των βιβλιοθηκών και ορισμένων δομικών στοιχείων τα οποία είναι πιθανό να χρησιμοποιηθούν στα πλαίσια των εργαστηριακών ασκήσεων του μαθήματος ΣΑΕ Ι. Περαιτέρω πληροφορίες για τα δομικά στοιχεία και τις βιβλιοθήκες του SIMULINK είναι διαθέσιμες μέσω της βοήθειας της MATLAB.

Σημείωση: η οργάνωση των βιβλιοθηκών και το γραφικό περιβάλλον παρουσιάζουν ορισμένες διαφορές μεταξύ των εκδόσεων 2.x και 3.x του SIMULINK, παρ' όλα αυτά οι βασικές λειτουργίες παραμένουν οι ίδιες. Οι πληροφορίες που δίνονται στη συνέχεια αφορούν την έκδοση 3.

6.2. Οι βασικές λειτουργίες του SIMULINK

Τα μοντέλα δυναμικών συστημάτων που κατασκευάζονται με το SIMULINK αποθηκεύονται ως αρχεία με την κατάληξη *.mdl*. Προκειμένου να δημιουργήσουμε ένα νέο μοντέλο ή να ανοίξουμε ένα αποθηκευμένο μοντέλο, στο παράθυρο εντολών της MATLAB κάνουμε αρχικά κλικ στο εικονίδιο  το οποίο ανοίγει τον browser των βιβλιοθηκών του SIMULINK. Από το παράθυρο του browser μπορούμε να δημιουργήσουμε ένα νέο μοντέλο, κάνοντας κλικ στο εικονίδιο  ή να ανοίξουμε ένα αποθηκευμένο μοντέλο, κάνοντας κλικ στο εικονίδιο  . Ένα τυπικό μοντέλο γραμμικού χρονικά αναλοίωτου συστήματος αυτομάτου ελέγχου μιας εισόδου μιας εξόδου με PID ελεγκτή παρουσιάζεται στο σχήμα που ακολουθεί (Εικόνα 6.1).

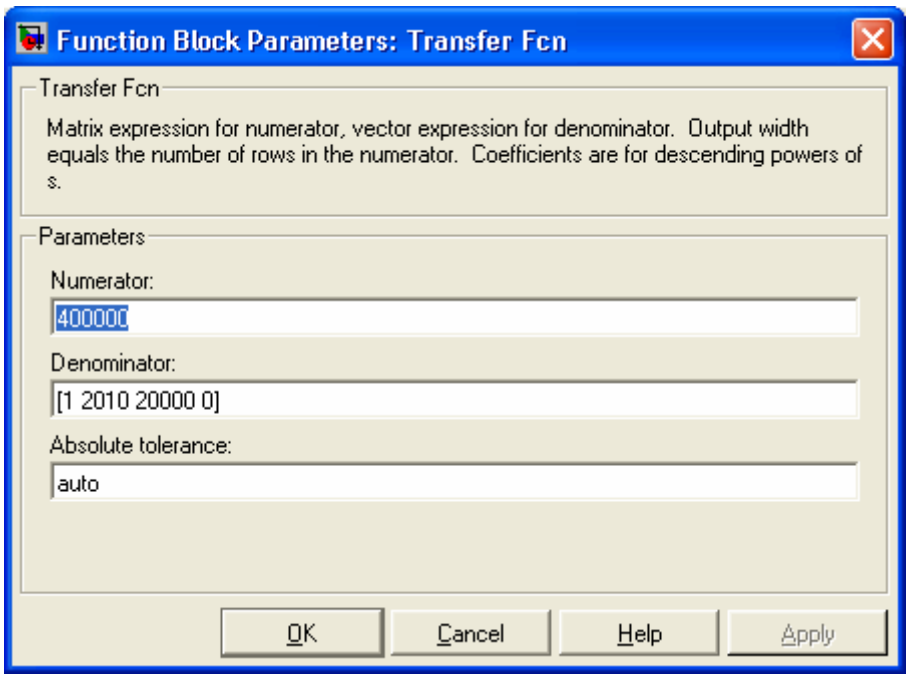


Εικόνα 6.1

Τα βασικά δομικά στοιχεία του μοντέλου είναι: α) το μοντέλο συνάρτησης μεταφοράς του ελεγχόμενου συστήματος (μπλοκ "servo model"), β) το μοντέλο του ελεγκτή PID (μπλοκ "PID Controller") γ) η είσοδος του συστήματος, σταθερά στην περίπτωση αυτή (μπλοκ "Constant", ανήκει στην κατηγορία των source blocks) και δ) ο αθροιστής (μπλοκ "Sum") ο οποίος κατασκευάζει το σήμα σφάλματος. Τα μπλοκ που περιέχουν το "K" πραγματοποιούν πολλαπλασιασμό με μια σταθερά (μπλοκ κέρδους, εδώ χρησιμοποιούνται για μετατροπή μονάδων). Τα αριθμημένα κυκλικά

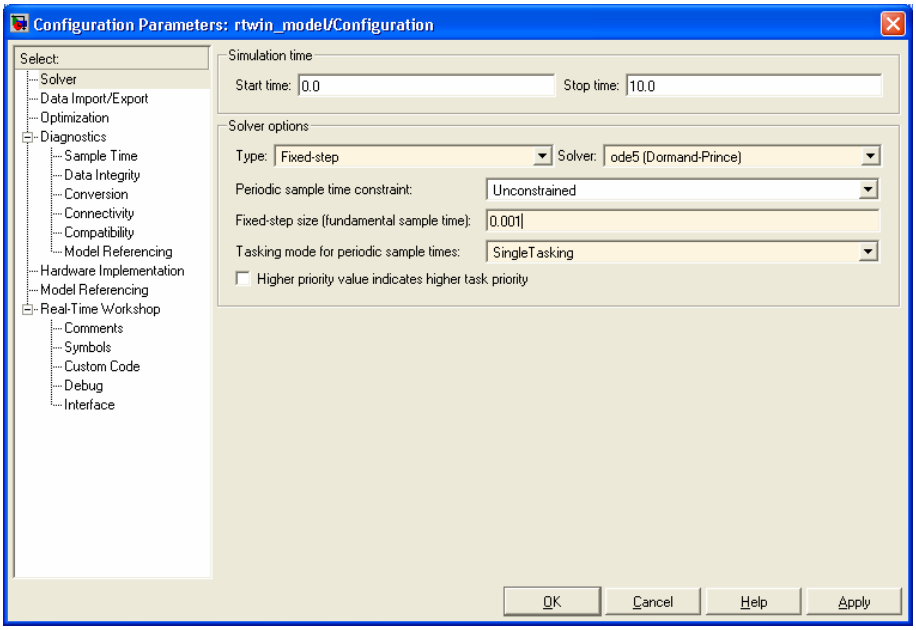
μπλοκ (ανήκουν στην κατηγορία των μπλοκ σημάτων και συστημάτων) αποτελούν τις θύρες εξόδου του μοντέλου (παρακολουθούμενα σήματα) προς το χώρο εργασίας της MATLAB, όπως θα δούμε αργότερα. Το μπλοκ με το όνομα "Saturation" είναι ένα μη γραμμικό στοιχείο κόρου.

Κάθε ένα από τα μπλοκ χαρακτηρίζεται από ορισμένες παραμέτρους. *Οι τιμές των παραμέτρων αυτών μπορούν να καθοριστούν κάνοντας διπλό κλικ πάνω στο μπλοκ*, οπότε ανοίγει ένα παράθυρο διαλόγου, όπως αυτό που φαίνεται στο σχήμα που ακολουθεί (Εικόνα 6.2) για το μπλοκ μοντέλου συνάρτησης μεταφοράς συστήματος μιας εισόδου μιας εξόδου:



Εικόνα 6.2

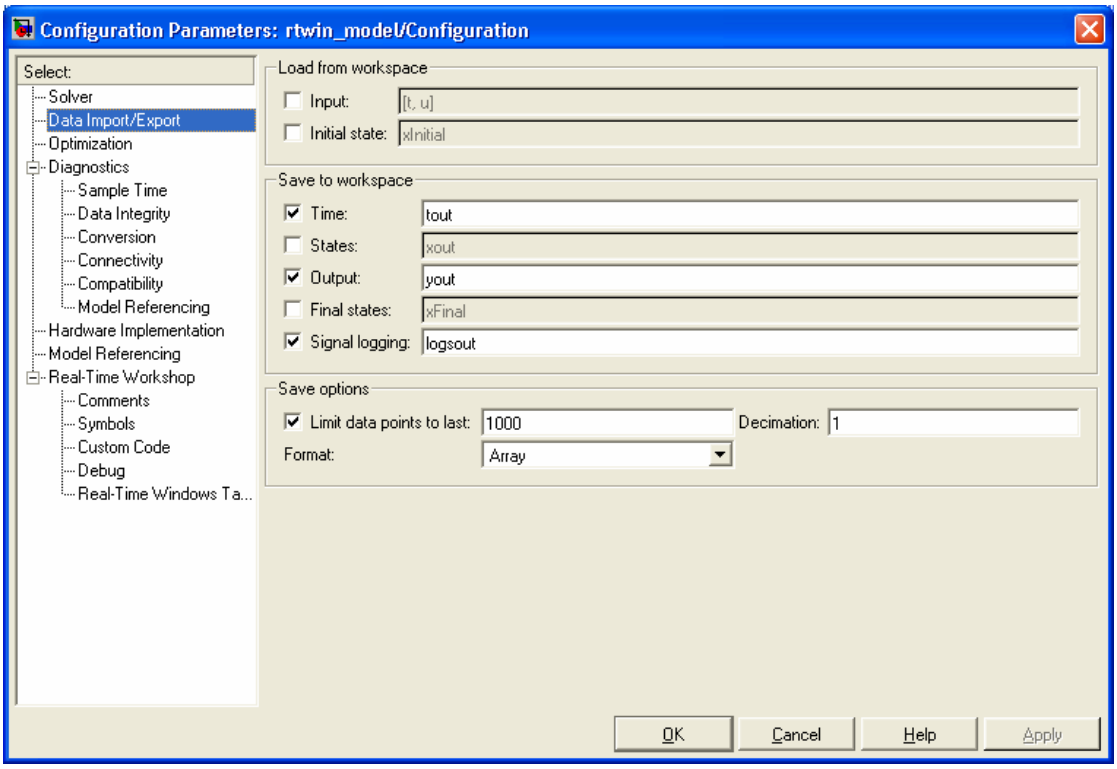
Μετά τον καθορισμό όλων των απαιτούμενων παραμέτρων για τα μπλοκ που περιλαμβάνονται στο μοντέλο, πρέπει να καθοριστούν οι παράμετροι της προσομοίωσης. Αυτό είναι δυνατό μέσω του μενού "Simulation" του παραθύρου του μοντέλου. Επιλέγοντας το μενού " *Configuration Parameters* …" εμφανίζεται το παράθυρο που φαίνεται στο σχήμα που ακολουθεί (Εικόνα 6.3):



Εικόνα 6.3

Μέσω της καρτέλας "Solver" καθορίζονται ο χρόνος έναρξης και λήξης της προσομοίωσης (σε δευτερόλεπτα), η μέθοδος ολοκλήρωσης που θα χρησιμοποιηθεί και ανάλογα με αυτήν οι απαιτούμενες παράμετροι (*σημείωση: τα συστήματα που μοντελοποιούνται στα πλαίσια του μαθήματος ΣΑΕ Ι είναι όλα συνεχούς χρόνου, επομένως και η επιλεγόμενη μέθοδος ολοκλήρωσης πρέπει να είναι συνεχούς χρόνου*). Μέσω της καρτέλας "Data Import/Export", που παρουσιάζεται στο σχήμα που ακολουθεί (Εικόνα 6.4), είναι δυνατή η επικοινωνία με το χώρο εργασίας της MATLAB. Πιο αναλυτικά:

- Μέσω των επιλογών "Load from workspace" καθορίζεται αν το μοντέλο θα δεχτεί είσοδο ή κάποια αρχική κατάσταση μέσω μεταβλητών του χώρου εργασίας (δεν έχει επιλεγεί κάτι τέτοιο στο συγκεκριμένο μοντέλο).



Εικόνα 6.4

- Μέσω των επιλογών "Save to workspace" καθορίζονται οι έξοδοι προς το χώρο εργασίας. Στο συγκεκριμένο μοντέλο έχει επιλεγεί η αποθήκευση στο χώρο εργασίας του χρόνου προσομοίωσης (διάνυσμα με τις χρονικές στιγμές της ολοκλήρωσης) ως μεταβλητή *tout* και των εξόδων του μοντέλου ως μεταβλητή *yout* (οι τιμές των τριών σημάτων που οδηγούνται στα sink blocks κατά τις χρονικές στιγμές της ολοκλήρωσης).

- Μέσω των επιλογών "Save options" καθορίζονται ορισμένες επιλογές για την αποθήκευση μεταβλητών στο χώρο εργασίας. Η επιλογή του "Limit data points to last:" έχει ως αποτέλεσμα την αποθήκευση μόνο του αριθμού των τελευταίων βημάτων της ολοκλήρωσης που εμφανίζεται στο παράθυρο (εξ' ορισμού 1000).

***Σημείωση**: αν δεν καθοριστούν παράμετροι προσομοίωσης, το SIMULINK χρησιμοποιεί τις εξ' ορισμού παραμέτρους (αυτό συμβαίνει γενικά με όλα τα μπλοκ που περιλαμβάνουν παραμέτρους).*

Μετά και τον καθορισμό των παραμέτρων της προσομοίωσης είναι δυνατή η εκτέλεση της. Αυτό μπορεί να γίνει είτε από το μενού “Simulation”, όπου επιλέγουμε “Start”, είτε κάνοντας κλικ στο εικονίδιο  . Η προσομοίωση σταματάει μόλις ο χρόνος φτάσει την τιμή που είχε τεθεί στην παράμετρο “Stop time”. Μπορούμε να διακόψουμε την προσομοίωση αν επιλέξουμε “Stop” από το μενού “Simulation” ή κάνουμε κλικ στο εικονίδιο  (όταν δεν τρέχει η προσομοίωση είναι απενεργοποιημένο).

Προκειμένου να δημιουργήσουμε ένα νέο μοντέλο SIMULINK εξ΄ αρχής, η διαδικασία είναι η ακόλουθη:

- κάνουμε κλικ στο εικονίδιο  του browser βιβλιοθηκών, οπότε ανοίγει ένα παράθυρο νέου μοντέλου (untitled)
- εισάγουμε τα επιθυμητά μπλοκ από τις κατάλληλες βιβλιοθήκες κάνοντας click-and-drag με το ποντίκι απ’ ευθείας στο παράθυρο του νέου μοντέλου (τα περιεχόμενα μιας βιβλιοθήκης εμφανίζονται κάνοντας διπλό κλικ πάνω στο εικονίδιο  του browser).
- Κάνουμε τις κατάλληλες διασυνδέσεις με click-and-drag από την έξοδο ενός μπλοκ στην είσοδο του άλλου
- Καθορίζουμε τις παραμέτρους των μπλοκ και της προσομοίωσης
- Αποθηκεύουμε το μοντέλο μέσω του μενού “File”

Μπορούμε να διαγράψουμε, να αντιγράψουμε, ή να κόψουμε και να επικολλήσουμε μπλοκ ή συνδέσεις -αφού πρώτα τα επιλέξουμε κάνοντας κλικ πάνω τους- με το συνήθη τρόπο. Μπορούμε επίσης να επιλέξουμε μια ομάδα μπλοκ και συνδέσεων κάνοντας click-and-drag με το ποντίκι και περιλαμβάνοντας τα.

6.3. Οι βασικές βιβλιοθήκες δομικών στοιχείων του SIMULINK

Στον πίνακα που ακολουθεί περιγράφονται ορισμένα από τα δομικά στοιχεία των βιβλιοθηκών του SIMULINK, οι οποίες χρησιμοποιούνται συνήθως κατά τη μοντελοποίηση γραμμικών χρονικά αναλλοίωτων συστημάτων συνεχούς χρόνου.

Βιβλιοθήκη	**Δομικό στοιχείο**	**Λειτουργία**
Sources - περιέχει μπλοκ τα οποία είναι *πηγές σημάτων* (δεν έχουν είσοδο, παράγουν ως έξοδο ένα σήμα)	Constant	Σταθερά
	Step	Βηματική συνάρτηση
	Ramp	Συνάρτηση αναρρίχησης
	Pulse generator	Γεννήτρια παλμών
	Random number	Γεννήτρια τυχαίου σήματος (κανονική κατανομή)
	Sine wave	Γεννήτρια ημιτόνου
	Signal generator	Γεννήτρια σημάτων (παράγει διάφορες κυματομορφές)
Sinks - περιέχει μπλοκ τα οποία είναι *στοιχεία «απορ-ρόφησης»* σημάτων (δεν έχουν έξοδο, δέχονται μόνο είσοδο)	Display	Οθόνη απεικόνισης τιμών
	Scope	Παλμογράφος
	Stop Simulation	Τερματισμός προσομοίωσης
	To Workspace	Αποθήκευση στο χώρο εργασίας
Continuous - περιέχει μπλοκ για τη μοντελοποίηση *γραμμικών συστημάτων συνεχούς χρόνου*	Derivative	Παραγώγιση
	Integrator	Ολοκλήρωση
	State space	Μοντέλο συστήματος στο χώρο κατάστασης
	Transfer function	Μοντέλο συστήματος συνάρ-τησης μεταφοράς
	Zero-pole	Μοντέλο συστήματος πόλων – μηδενικών
Nonlinear - περιέχει μπλοκ που μοντελοποιούν *μη γραμ-μικά στοιχεία*	Saturation	Στοιχείο κόρου
	Manual Switch	Χειροκίνητος διακόπτης
	Switch	Διακόπτης
	Quantizer	Κβαντιστής σήματος

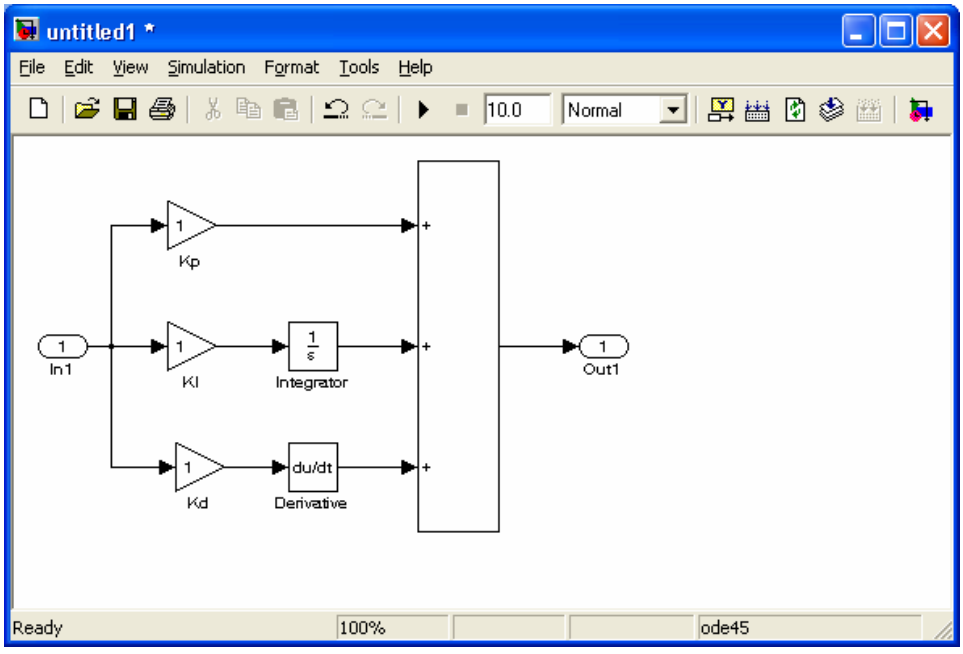
Βιβλιοθήκη	**Δομικό στοιχείο**	**Λειτουργία**
Math - Περιέχει μπλοκ που μοντελοποιούν *μαθημα-τικές πράξεις και συναρτήσεις*	Abs	Απόλυτη τιμή
	Gain	Κέρδος
	Math function	Διάφορες μαθηματικές συναρτήσεις
	Matrix Gain	Πίνακας κερδών
	MinMax	Ελάχιστο ή μέγιστο
	Product	Πολλαπλασιασμός ή διαίρεση
	Rounding Function	Συνάρτηση στρογγύλευσης
	Sign	Εύρεση προσήμου
	Slider gain	Μεταβλητό κέρδος
	Sum	Άθροιση ή αφαίρεση
	Trigonometric function	Τριγωνομετρικές συναρτήσεις
Signals and Systems - Περιέχει στοιχεία *διασύνδεσης σημάτων και συστημάτων*	In1	Θύρα εισόδου υποσυστήματος ή μοντέλου
	Demux	Αποπλέκτης σημάτων
	Mux	Πολυπλέκτης σημάτων

	Terminator	Τερματισμός ασύνδετων σημάτων (δέχεται μόνο είσοδο)
	Out1	Θύρα εξόδου υποσυστήματος ή μοντέλου
Control Systems Toolbox - Περιέχει στοιχεία μοντελοποίησης *συστημάτων ελέγχου*	LTI System	Μοντελοποίηση γραμμικού χρονικά αναλλοίωτου συστήματος με διάφορους τρόπους (μέσω συνάρτησης μεταφοράς, στο χώρο κατάστασης, αναπαράσταση πόλων – μηδενικών)

Υπενθυμίζεται ότι η οργάνωση των βιβλιοθηκών του SIMULINK όπως παρουσιάστηκε στον προηγούμενο πίνακα αναφέρεται στην *έκδοση 3*.

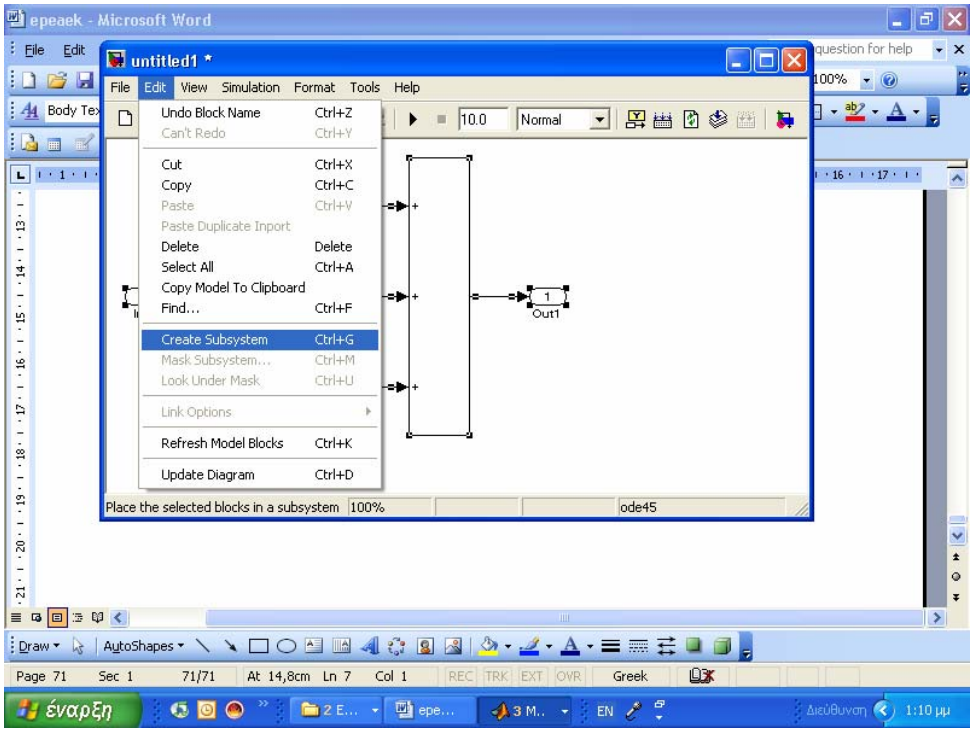
6.4. Παράδειγμα δημιουργίας απλού μοντέλου SIMULINK .

Αρχικά φτιάχνουμε το *PID ελεγκτή* στην γενική του μορφή. Από την εργαλειοθήκη *Simulink-continuous* επιλέγω τα μπλοκ *Integrator* και *derivative*. Επίσης από την εργαλειοθήκη *Simulink-math operations* επιλέγω το μπλοκ *gain* το οποίο το χρησιμοποιώ 3 φορές μέσα στον PID ελεγκτή. Από τις εργαλειοθήκες *Simulink-sinks* και *sources* επιλέγω τα μπλοκ *out1* και *in1* αντίστοιχα. Και τέλος από την εργαλειοθήκη *Simulink-math* operations επιλέγω το μπλοκ sum, οπότε το μοντέλο του *PID ελεγκτή* στην γενική του μορφή γίνεται (Εικόνα 6.5):

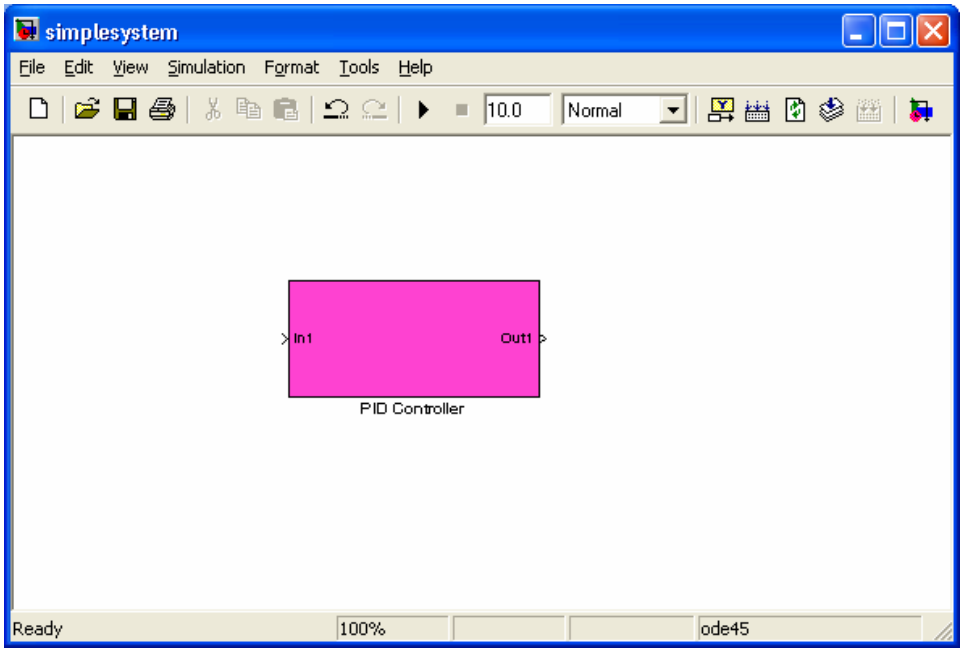


Εικόνα 6.5

Στη συνέχεια (Εικόνα 6.6, Εικόνα 6.7) από το μενού *edit-Create* Subsystem δημιουργούμε το μπλοκ του *PID* ελεγκτή:

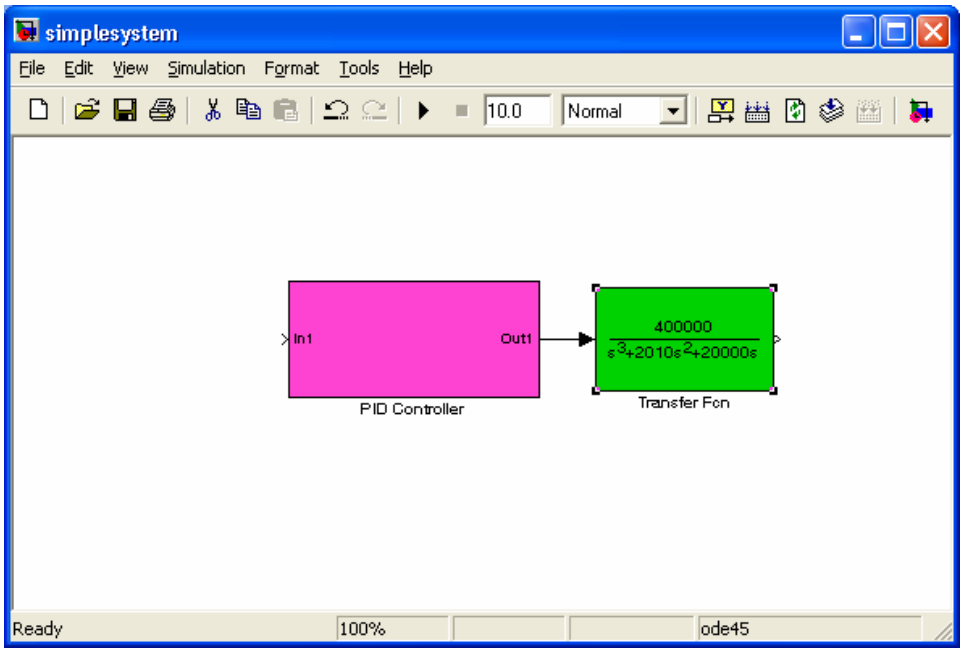


Εικόνα 6.6



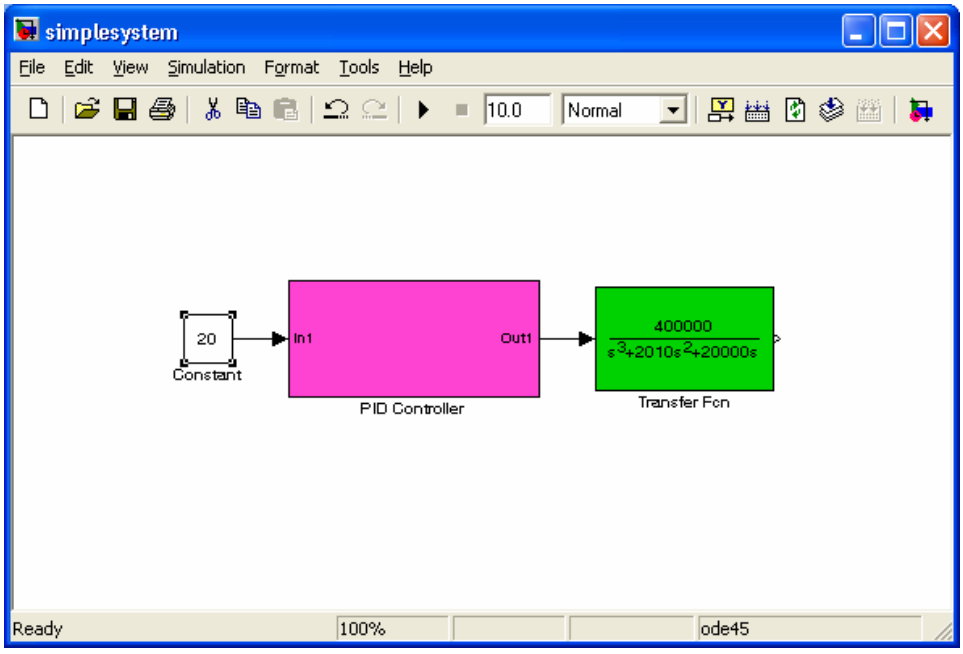
Εικόνα 6.7

Στη συνέχεια από την εργαλειοθήκη *Simulink- continuous* επιλέγω το μπλοκ *Transfer- Fcn* για να δημιουργήσω τη συνάρτηση μεταφοράς (Εικόνα 6.8).



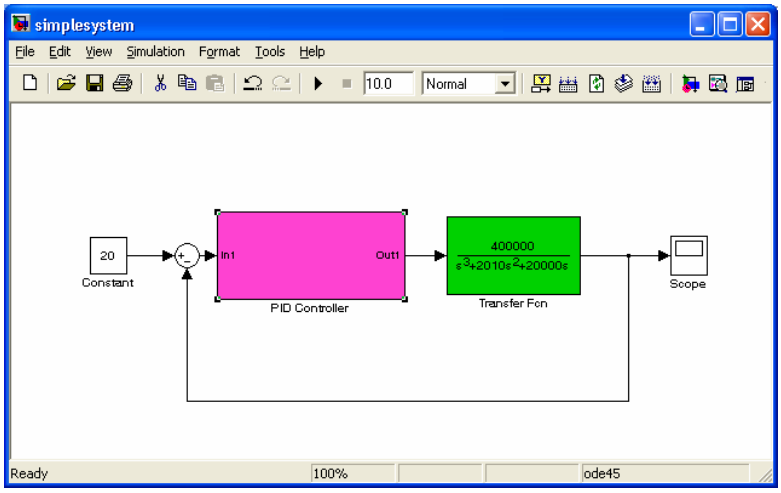
Εικόνα 6.8

Έπειτα από την εργαλειοθήκη *Simulink- sources* επιλέγω το μπλοκ *constant*, για να δημιουργήσω την είσοδο του συστήματος (Εικόνα 6.9).



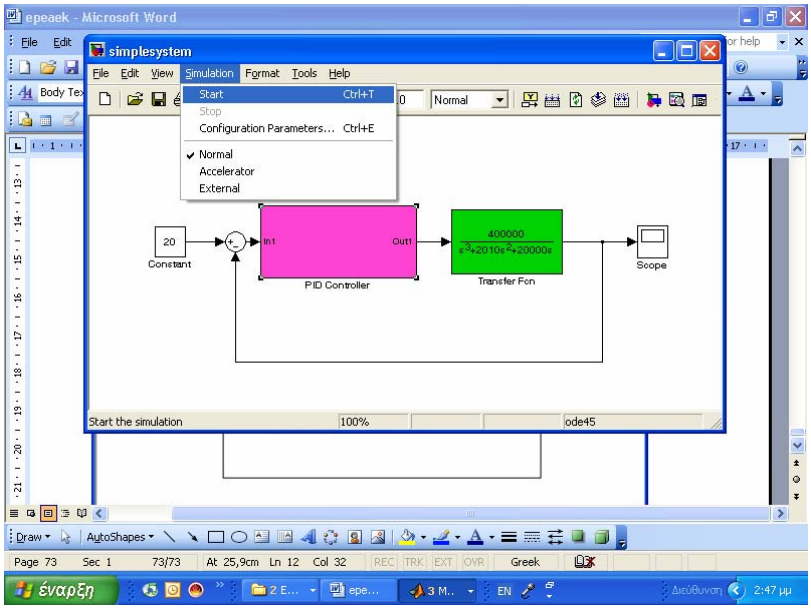
Εικόνα 6.9

Στη συνέχεια ολοκληρώνω τη δημιουργία του συστήματος βάζοντας την ανατροφοδότηση με τη βοήθεια και του μπλοκ sum από την εργαλειοθήκη *Simulink-math operations*. Τέλος από την εργαλειοθήκη *Simulink- sinks* επιλέγω το μπλοκ *scope (παλμογράφος)* το οποίο μας επιτρέπει να απεικονίσουμε τα αποτελέσματα της εξόδου (Εικόνα 6.10).

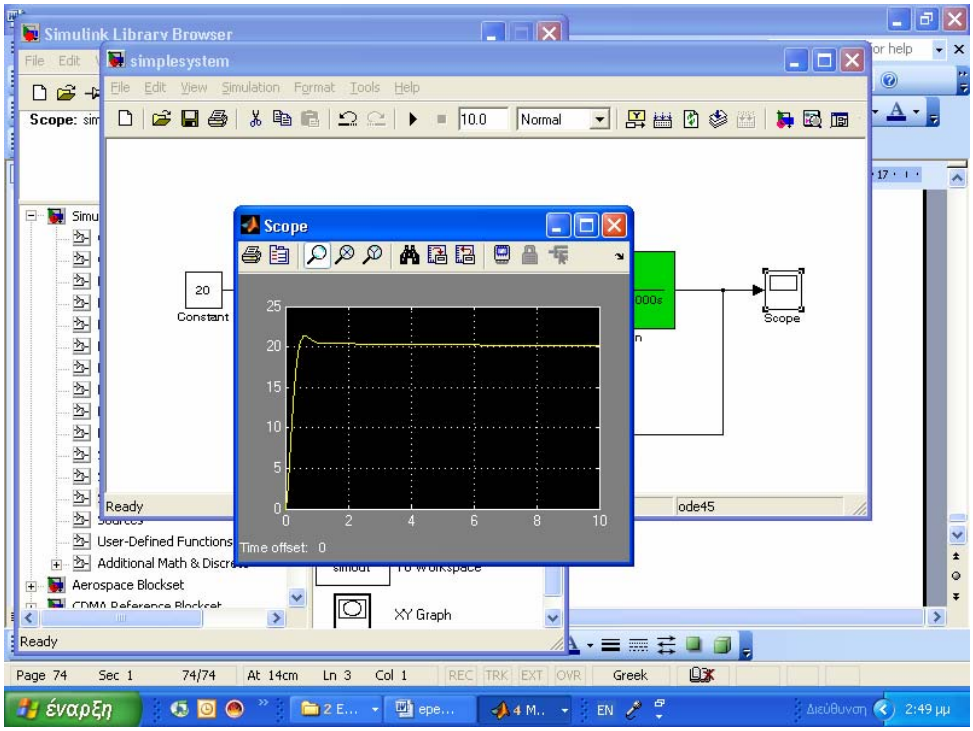


Εικόνα 6.10

Αν τώρα αντικαταστήσω τις παραμέτρους $K_p = 0.25$, $K_I = 0.03$ και $K_d = 0$ στο μπλοκ (*subsystem) PID Controller* και από το μενού *Simulation* επιλέξω *Start* το διάγραμμα *simulink* θα δώσει αποτελέσματα στην έξοδο μέσω του μπλοκ *παλμογράφου (scope)* (Εικόνα 6.11, Εικόνα 6.12).

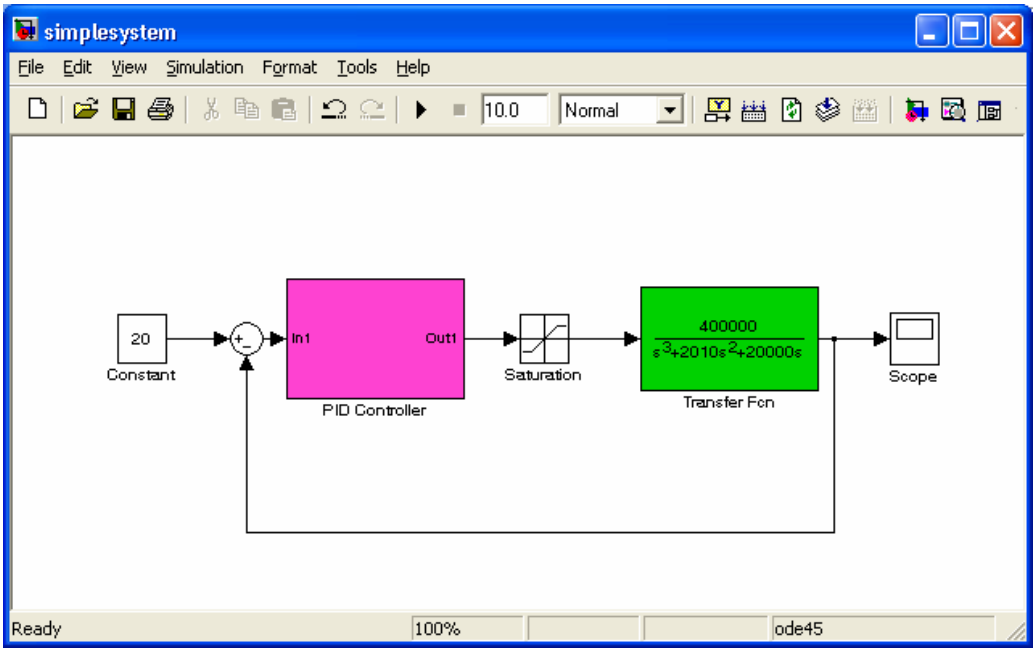


Εικόνα 6.11

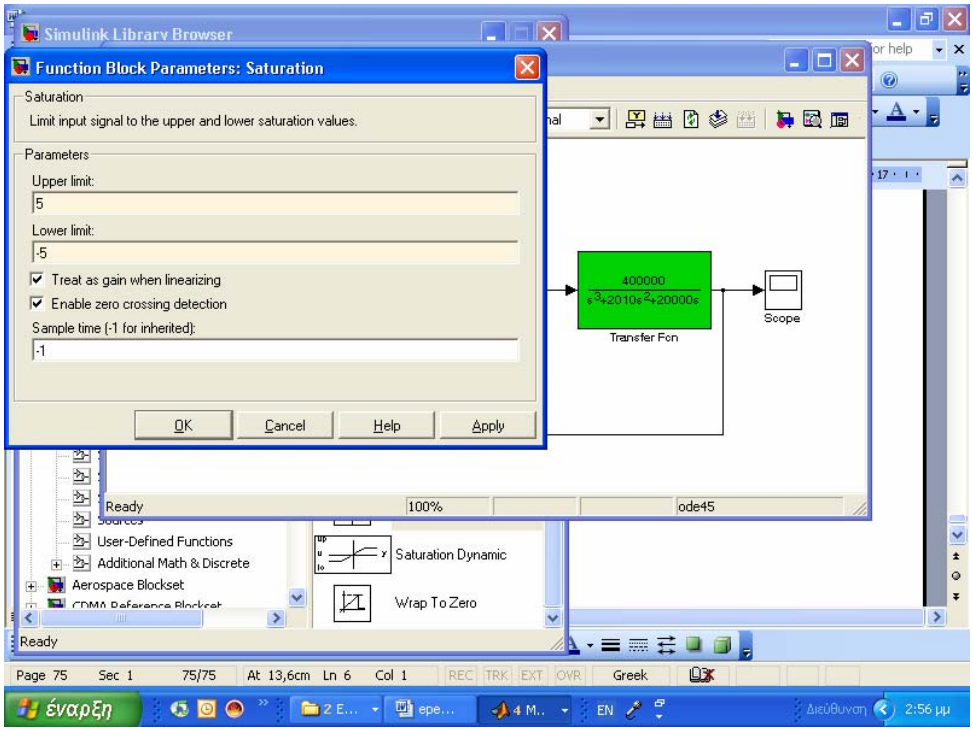


Εικόνα 6.12

Αν τώρα θέλουμε να δώσουμε στο σύστημα μη γραμμικά χαρακτηριστικά τότε από την εργαλειοθήκη *Simulink-discontinuities* επιλέγω το μπλοκ *Saturation (ρεύμα κόρου)* (Εικόνα 6.13, Εικόνα 6.14) το οποίο δεν επιτρέπει στην τάση ελέγχου να ξεφύγει από τις επιθυμητές τιμές (στην περίπτωση μας η μέγιστη τάση ελέγχου είναι $\pm 5V$).

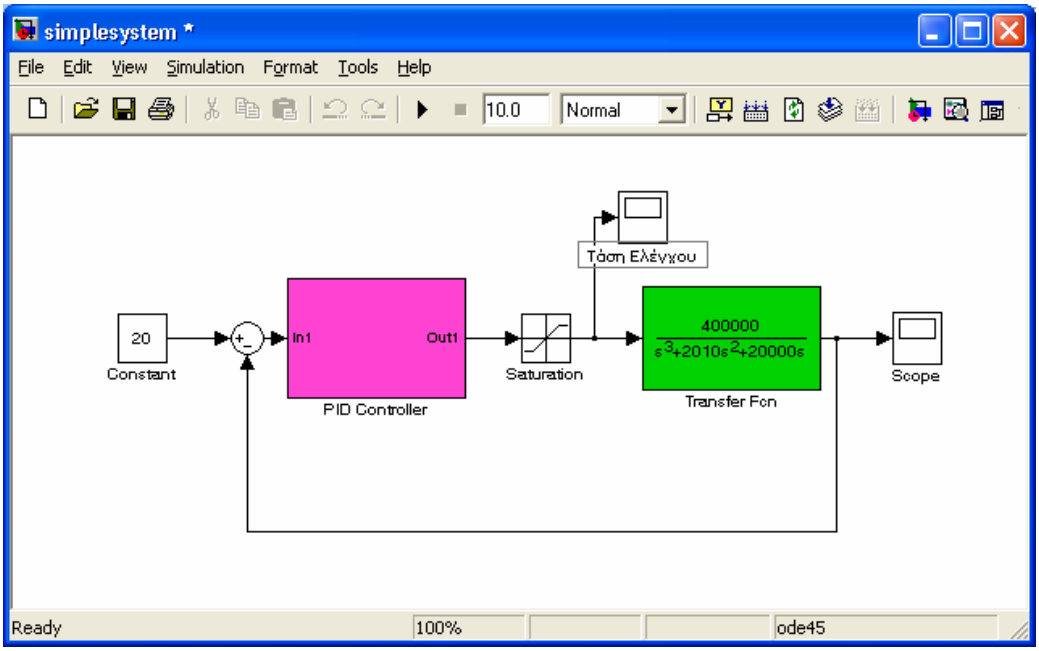


Εικόνα 6.13

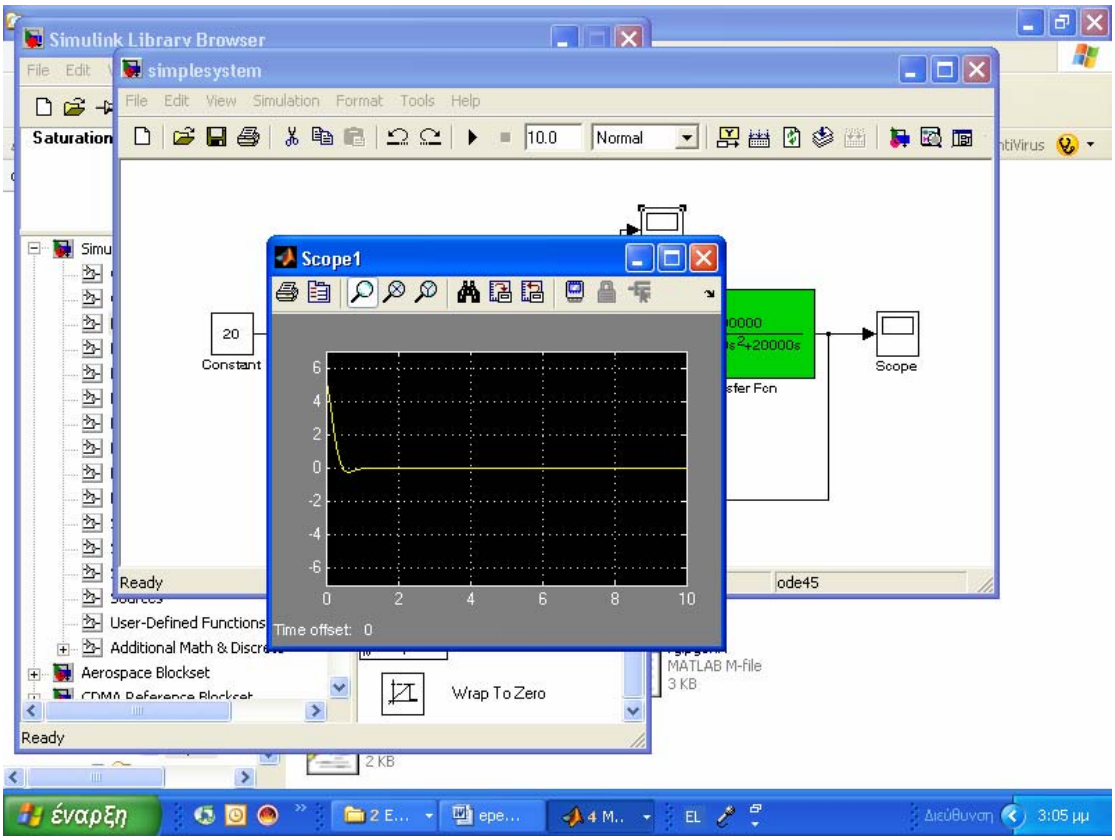


Εικόνα 6.14

Χρησιμοποιώντας πάλι ένα μπλοκ *scope (παλμογράφος)* ολοκληρώνουμε το νέο σύστημα και από το μενού *Simulation* επιλέξω πάλι *Start,* οπότε το διάγραμμα *simulink* θα δώσει αποτελέσματα (Εικόνα 6.15, Εικόνα 6.16) στην έξοδο μέσω των μπλοκ *παλμογράφου (scope)* (για την είσοδο που επιθυμούμε και για την τάση ελέγχου).



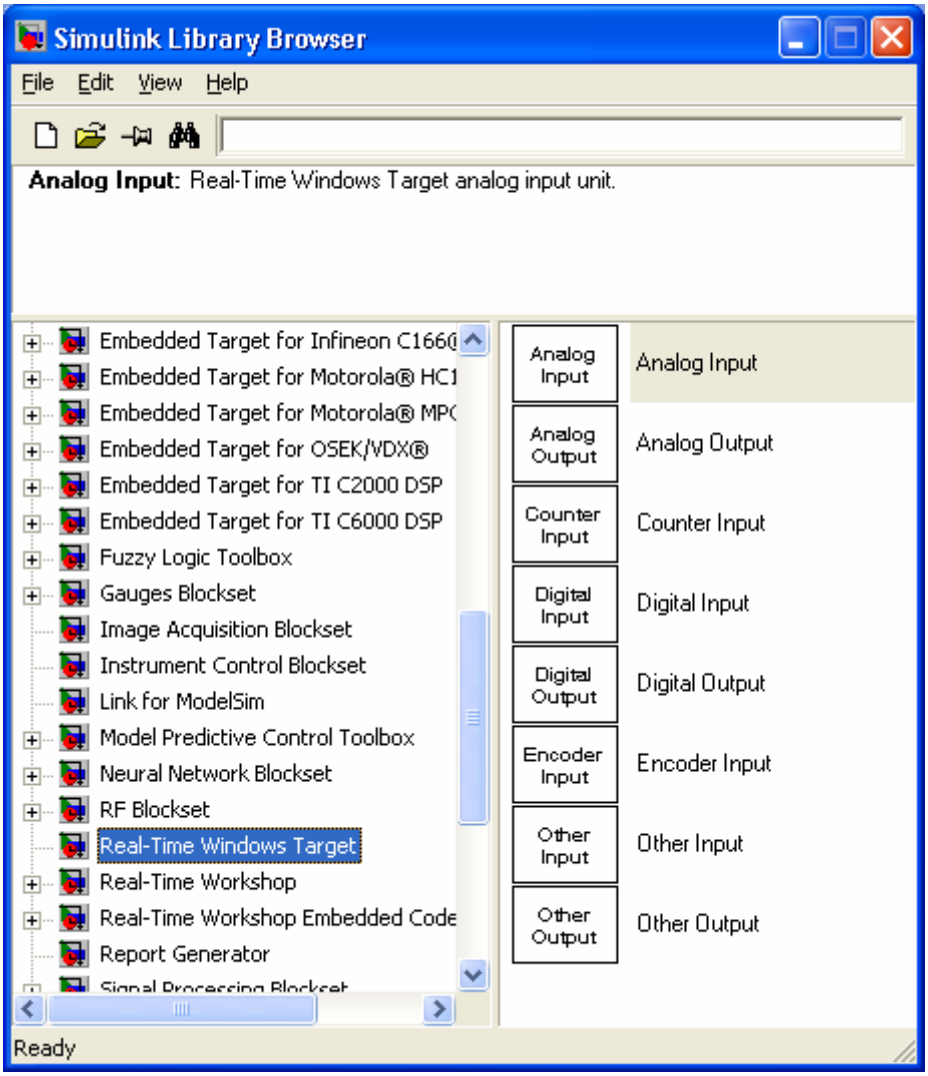
Εικόνα 6.15



Εικόνα 6.16

6.5. Εργαλειοθήκη Real-Time Windows Target

Αν μπούμε μέσα στον *Simulink Library Browser* και επιλέξουμε *Real-Time Windows Target* έχουμε (Εικόνα 6.17)



Εικόνα 6.17

Τα μπλοκ του *Real-Time Windows Target* μας επιτρέπουν τη λειτουργία μερικών πειραματικών διατάξεων (δεξαμενή, Αντεστραμένο εκκρεμές κλπ) σε πραγματικό χρόνο (Real Time Control) μέσω του προγράμματος MATLAB-SIMULINK.

Η πρόσκτηση δεδομένων και ο έλεγχος κάθε πειραματικής διάταξης γίνονται μέσω ενός Η/Υ στον οποίο είναι εγκατεστημένη η κάρτα πρόσκτησης δεδομένων. Η κάρτα αυτή συνδέεται με την πειραματική διάταξη (έξοδος συστήματος) και με τον Η/Υ (είσοδος συστήματος). Τα δεδομένα (είσοδος συστήματος) δίνονται μέσω ενός μοντέλου που έχει κατασκευαστεί σε Simulink και το οποίο, με χρήση των κατάλληλων μπλοκ του *Real-Time Windows Target*, συνδέεται με τις εισόδους και τις εξόδους της κάρτας πρόσκτησης δεδομένων. Με τον τρόπο αυτό επιτυγχάνουμε έλεγχο των πειραματικών διατάξεων σε πραγματικό χρόνο (τα μπλοκ που χρησιμοποιούμε στην περίπτωση μας είναι τα μπλοκ *Analog Input* και *Analog Output*. Όταν φτιαχτεί το μοντέλο σε Simulink στην συνέχεια εκτελείται το πείραμα σε

πραγματικό χρόνο. Θα δείξουμε στη συνέχεια τα βήματα που πρέπει να γίνουν για την εκτέλεση ενός μοντέλου πραγματικού χρόνου σε Simulink για την περίπτωση του ελέγχου στάθμης υγρού δεξαμενής. Τα βήματα αυτά είναι:

Ανοίγουμε το μοντέλο *realtimelast1.mdl*.
Δίνουμε την επιθυμητή στάθμη σε εκατοστά (επιτρεπτές τιμές από 0 ως 15, συνίσταται να δοθεί μια τιμή στην περιοχή 5-10 εκατοστά) στο μπλοκ *h1d (cm)*.
Δίνουμε τις παραμέτρους του ελεγκτή στο μπλοκ PI controller (η παράμετρος K_D είναι μηδέν για PI ελεγκτή).
Καλούμε τον υπεύθυνο της άσκησης.
Από το μενού *Simulation* επιλέγουμε *Connect to target* (η επιλογή *External* του ίδιου μενού είναι επίσης ενεργή).
Μόλις η προηγούμενη επιλογή ενεργοποιηθεί, από το μενού *Simulation* επιλέγουμε *Start real-time code* και αρχίζει η λειτουργία του μοντέλου πραγματικού χρόνου. Ο χρόνος λειτουργίας του μοντέλου είναι ρυθμισμένος στα 120sec.
Ανοίγουμε τη διάταξη (διακόπτης του τροφοδοτικού στη θέση I).
Μόλις ολοκληρωθεί η εκτέλεση του πειράματος, κλείνουμε τη διάταξη.
Τα βήματα αυτά είναι λίγο πολύ ίδια σε όλες τις πειραματικές διατάξεις.
Πριν φτιάξουμε το μοντέλο σε Simulink πρέπει να κάνουμε εγκατάσταση *του Real-Time Windows Target Kernel*.
Στο παράθυρο της MATLAB πληκτρολογούμε

```
rtwintgt -install
```

Στη συνέχεια εμφανίζεται το μήνυμα

```
You are going to install the Real-Time Windows Target kernel.
Do you want to proceed? [y] :
```

Πληκτρολογούμε *y* και εμφανίζεται το μήνυμα

```
The Real-Time Windows Target kernel has been successfully
installed.
```

Για να τρέξουμε ένα μοντέλο πραγματικού χρόνου σε MATLAB-SIMULINK χρειαζόμαστε στον υπολογιστή μας εγκαταστημένο ένα *compiler* σε *Visual C++ 6.*

Αυτός δημιουργεί τα *source files* και *header files* που είναι απαραίτητα για να «τρέξει» μια εφαρμογή σε πραγματικό χρόνο. Γι' αυτό αφού πρώτα εγκαταστήσουμε τον *compiler* της *Visual C++ 6* στον υπολογιστή μας στη συνέχεια στο παράθυρο της MATLAB πληκτρολογούμε

```
>> mex –setup
```

Οπότε εμφανίζεται το μήνυμα

```
Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n?
```

Επιλέγουμε *y* και έχουμε

```
Select a compiler:
[1] Digital Visual Fortran version 6.0 in C:\Program Files\Microsoft Visual Studio
[2] Lcc C version 2.4 in C:\MATLAB701\sys\lcc
[3] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler:
```

Επιλέγουμε *3* και έχουμε

```
Please verify your choices:

Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Program Files\Microsoft Visual Studio

Are these correct?([y]/n):
```

Επιλέγουμε *y* και έχουμε

```
Are these correct?([y]/n): y
```

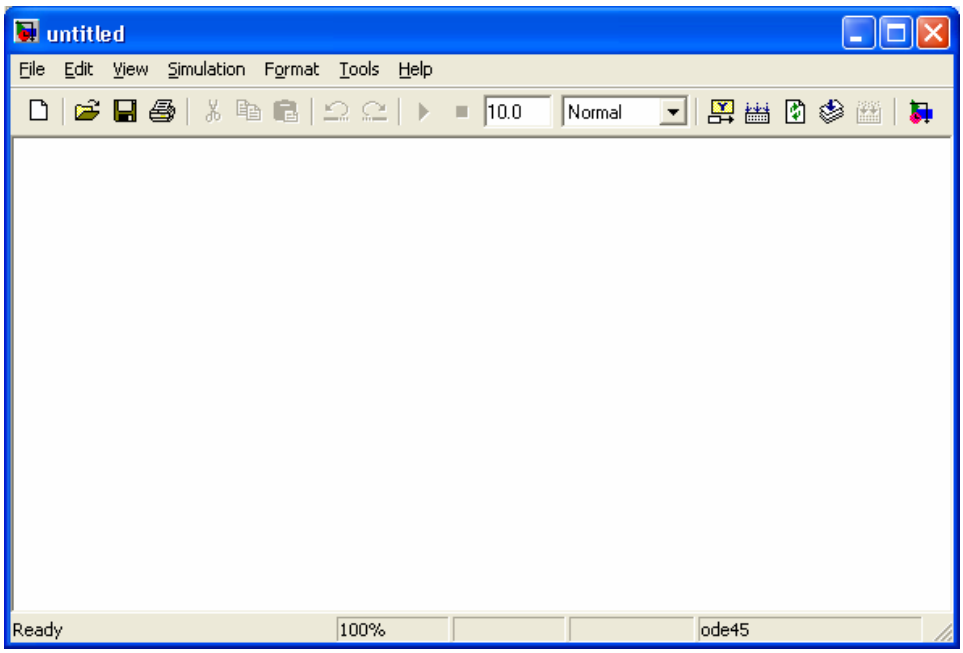
Try to update options file: C:\Documents and Settings\ÍåêôÜñéïòÁñíáïõôÜêçò\Application Data\MathWorks\MATLAB\R14\mexopts.bat
From template: C:\MATLAB701\BIN\WIN32\mexopts\msvc60opts.bat

Done . . .

Οπότε έγινε η εγκατάσταση του *Compiler* στη MATLAB.

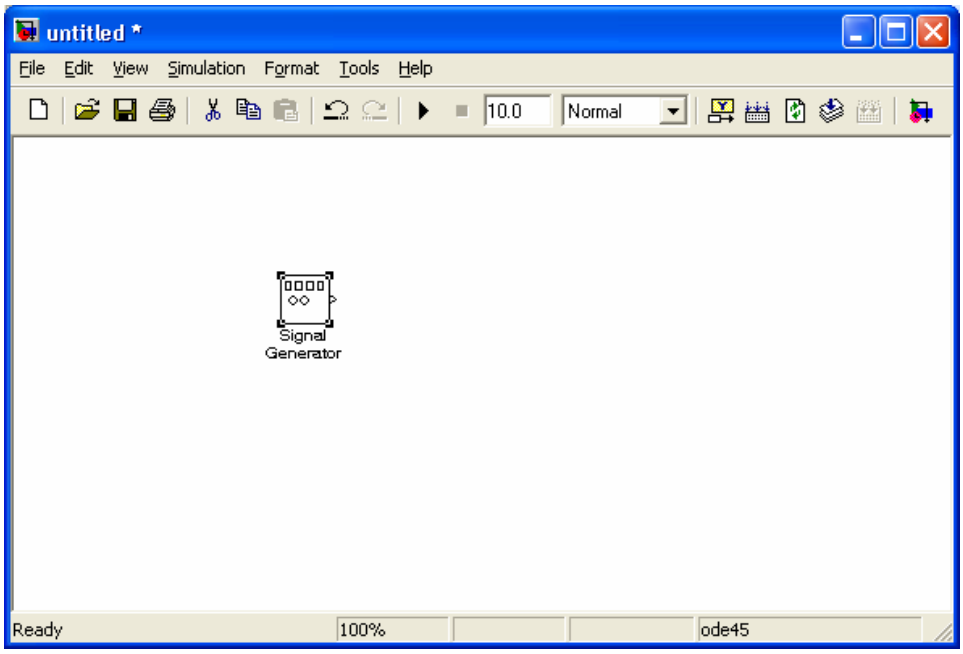
Στη συνέχεια θα δείξουμε τη δημιουργία ενός απλού μοντέλου σε Simulink.

Στον *Simulink Library Browser* επιλέγουμε *File-New* για να δημιουργήσουμε ένα καινούριο μοντέλο Simulink (Εικόνα 6.18).



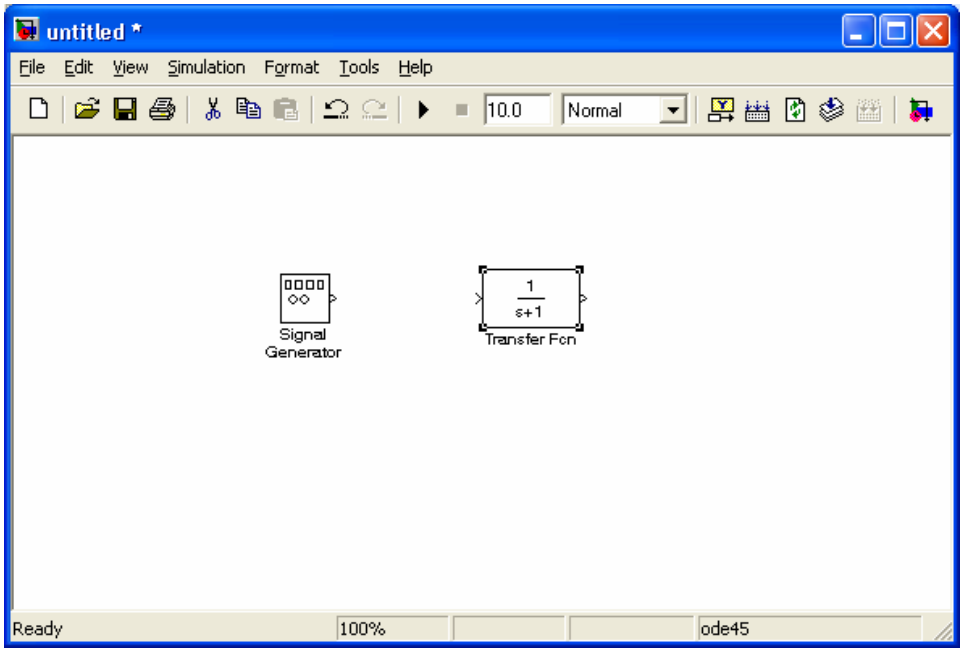
Εικόνα 6.18

Στην συνέχεια πάλι στον *Simulink Library Browser* κάνω διπλό κλικ στο *Simulink* και μετά κλικ στο *Sources*. Επιλέγω το μπλοκ *Signal Generator* και το τοποθετώ στο κενό αρχείο Simulink που άνοιξα (Εικόνα 6.19).



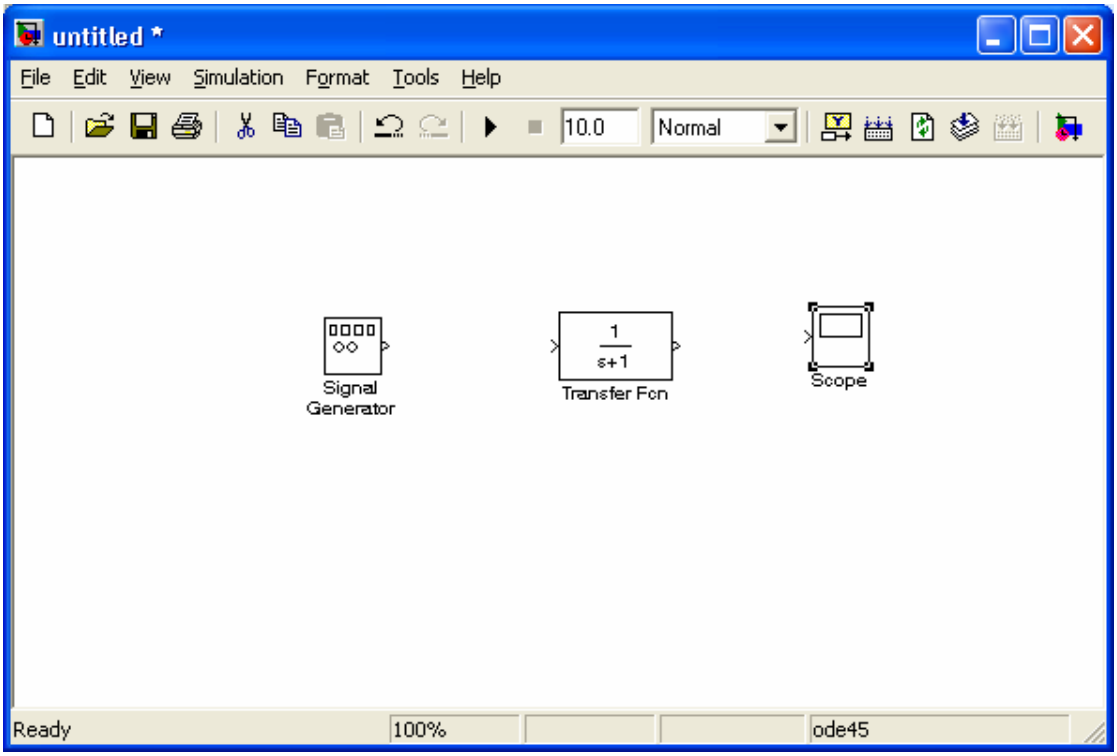
Εικόνα 6.19

Στη συνέχεια κάνω κλικ στο *Continuous*. . Επιλέγω το μπλοκ *Transfer Fcn* και το τοποθετώ στο κενό αρχείο Simulink που άνοιξα (Εικόνα 6.20).



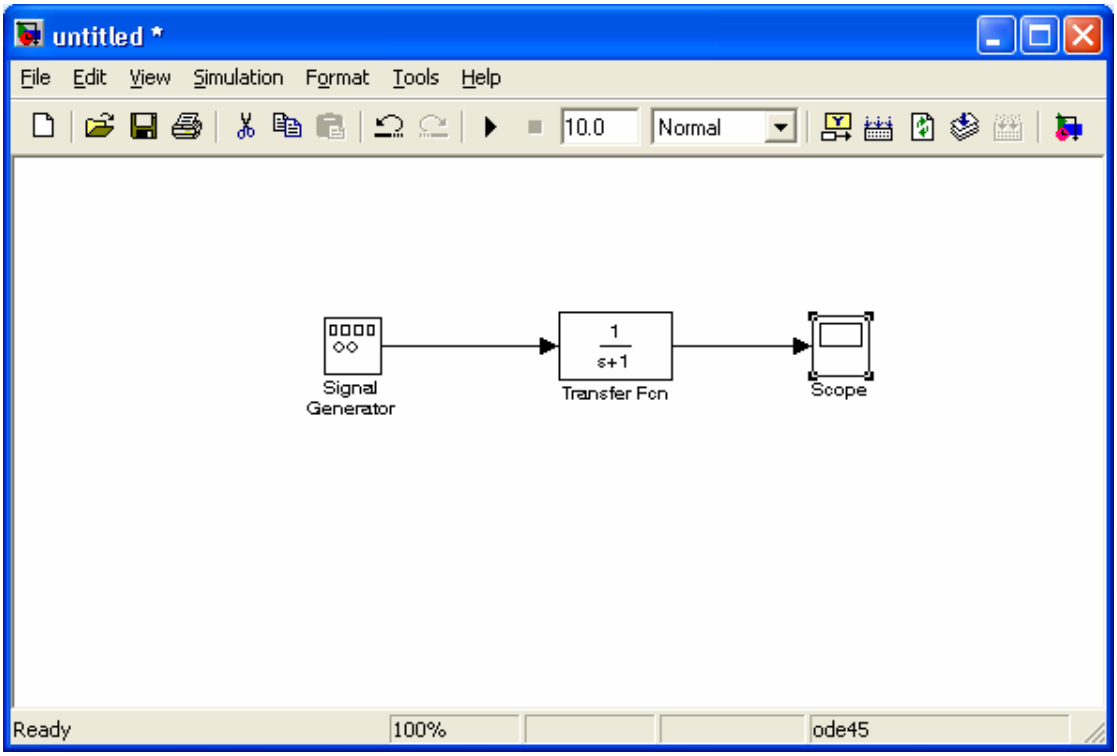
Εικόνα 6.20

Στη συνέχεια κάνω κλικ στο *Sinks*. . Επιλέγω το μπλοκ *Scope* και το τοποθετώ στο κενό αρχείο Simulink που άνοιξα (Εικόνα 6.21).



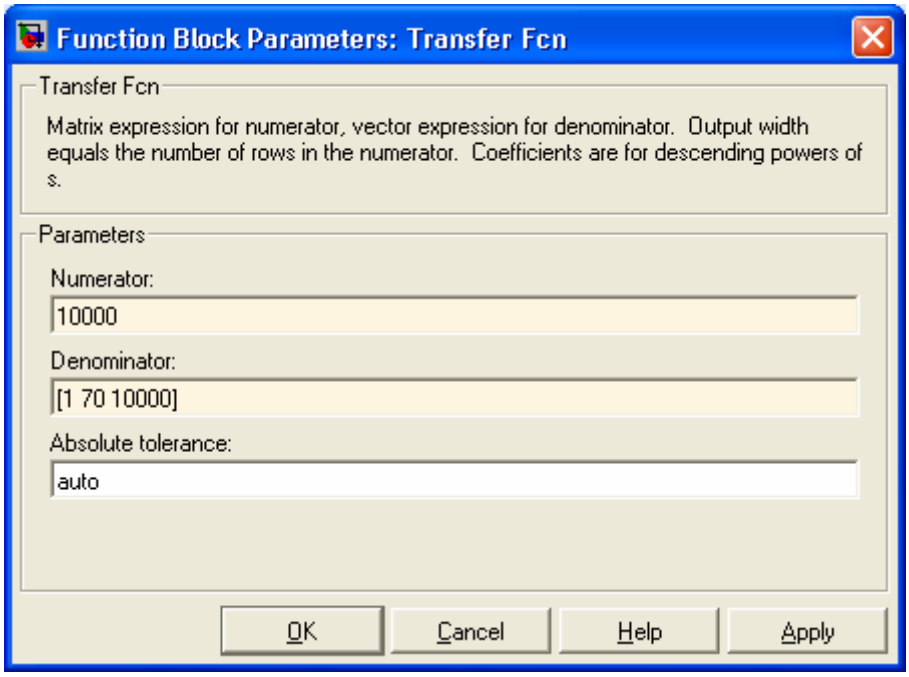
Εικόνα 6.21

Στη συνέχεια συνδέω τα μπλοκ *Signal Generator*, *Transfer Fcn* και *Scope (*Εικόνα 6.22).

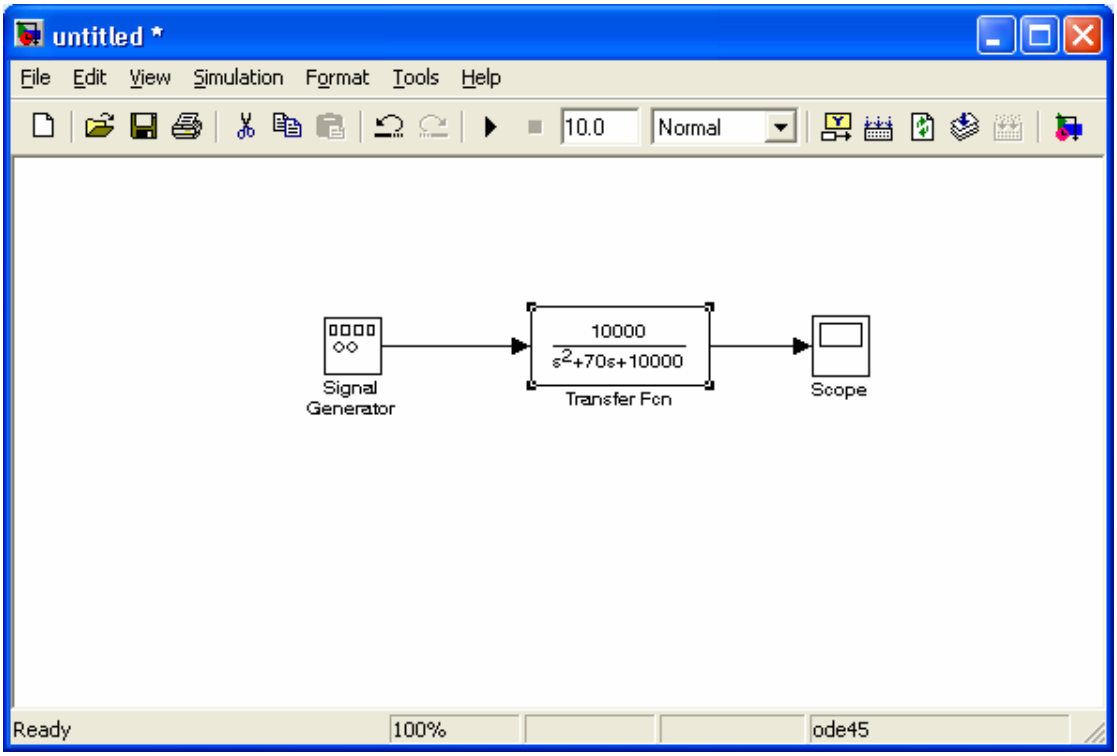


Εικόνα 6.22

Έπειτα κάνω διπλό κλικ στο μπλοκ *Transfer Fcn* και τοποθετώ τη συνάρτηση $\frac{10000}{s^2+70s+10000}$ (Εικόνα 6.23, Εικόνα 6.24)

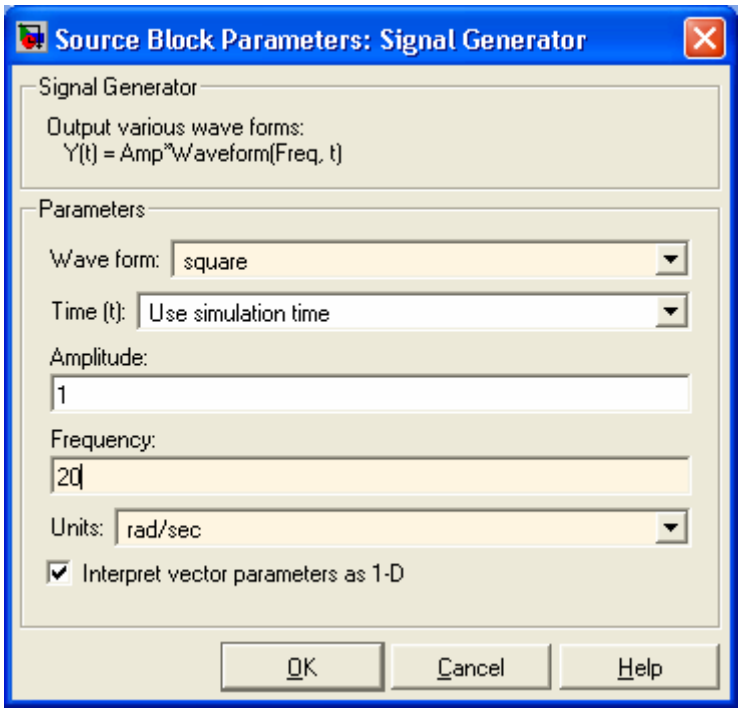


Εικόνα 6.23



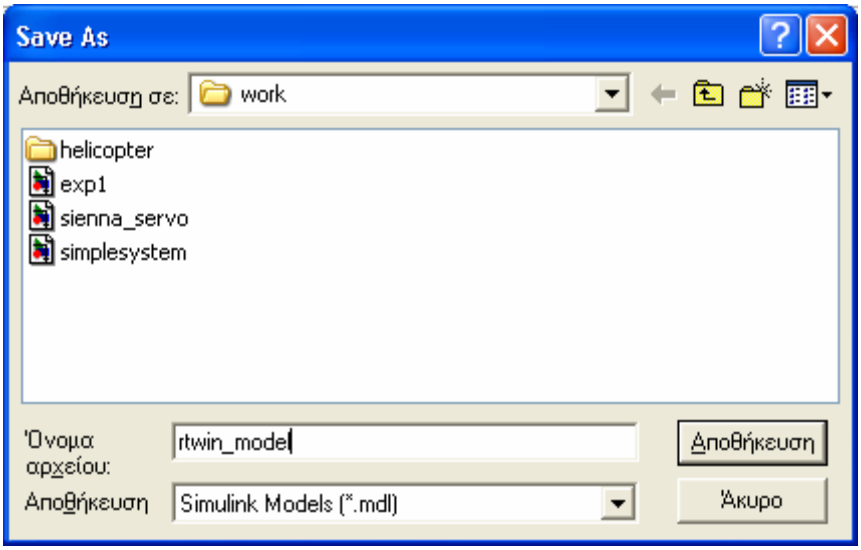
Εικόνα 6.24

Έπειτα κάνω διπλό κλικ στο μπλοκ *Signal Generator* και τοποθετώ τις παρακάτω παραμέτρους (Εικόνα 6.25)

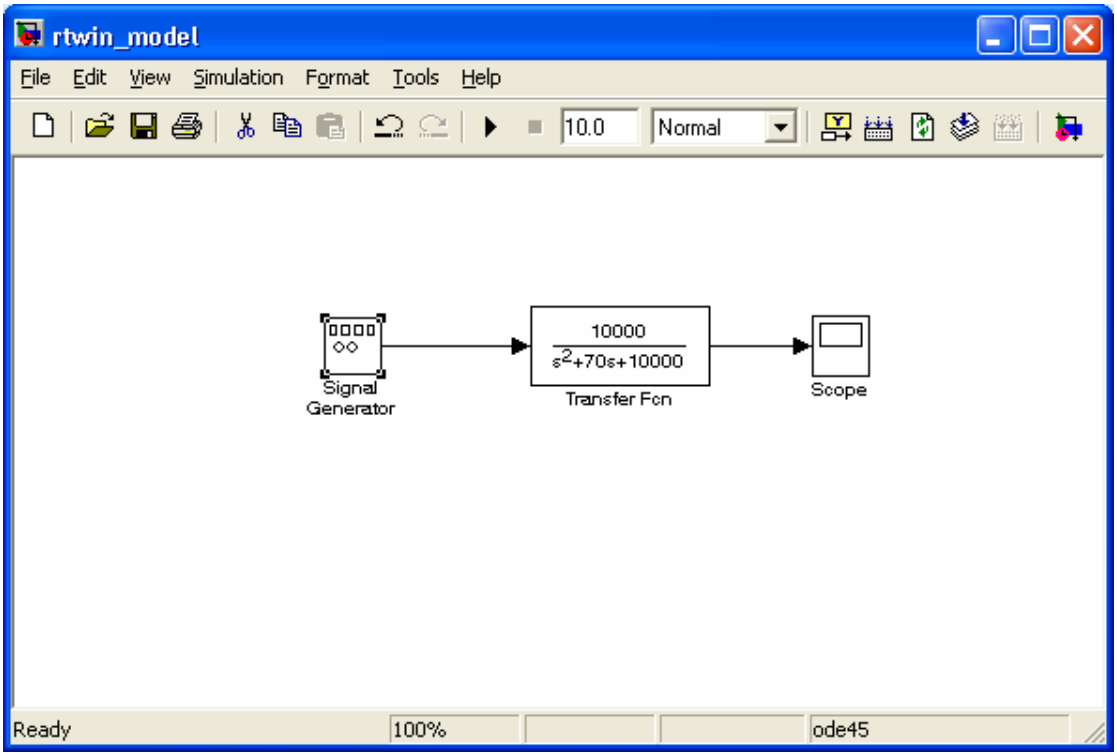


Εικόνα 6.25

Στη συνέχεια επιλέγω *File-Save As* και ονομάζω το μοντέλο *rtwin_model* (Εικόνα 6.26, Εικόνα 6.27).



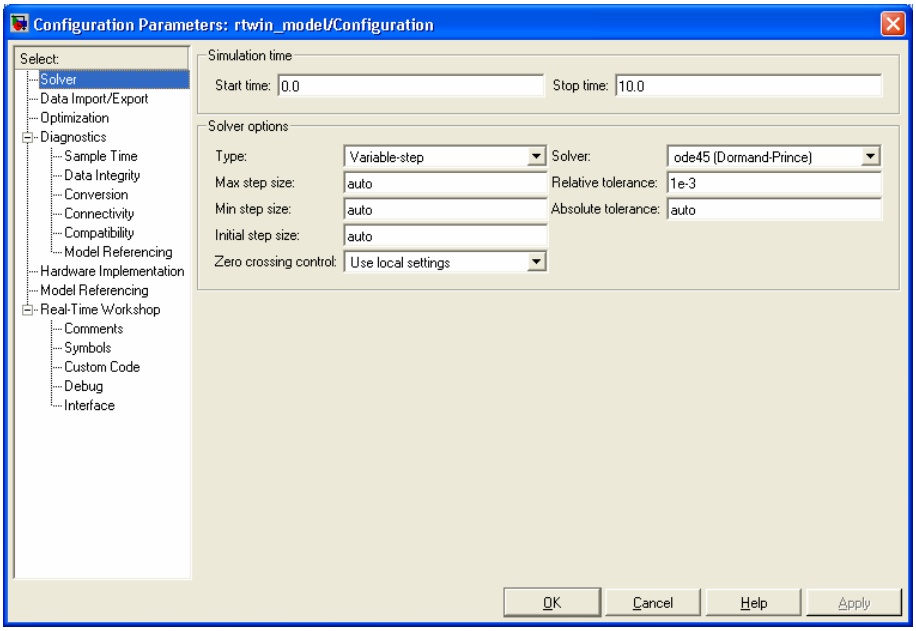
Εικόνα 6.26



Εικόνα 6.27

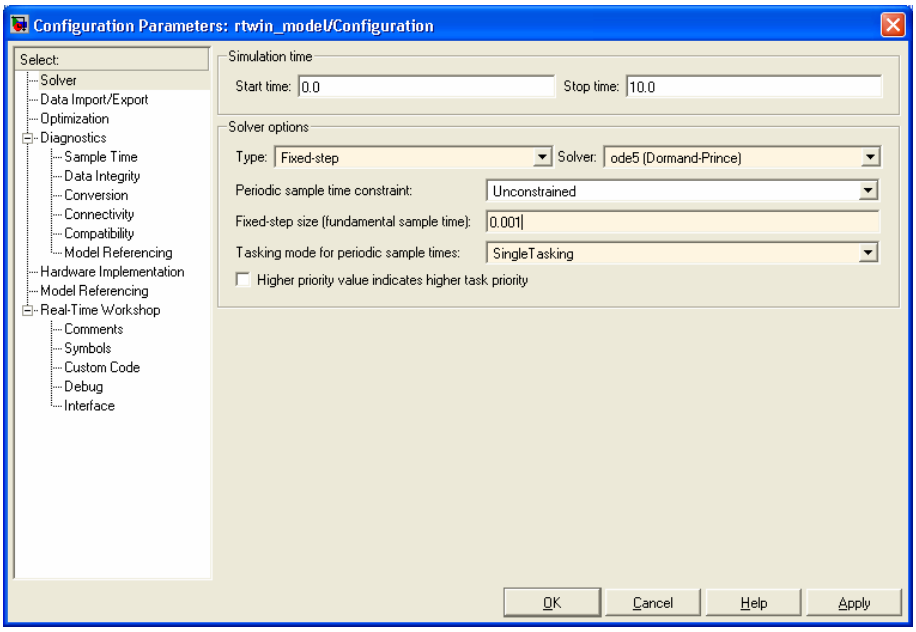
Στη συνέχεια από το *Simulation* μενού κάνω κλικ στο *Configuration Parameters*.

Στο *Start time* box βάζω 0.0 και στο *Stop time* box βάζω 10.0 seconds (Εικόνα 6.28).



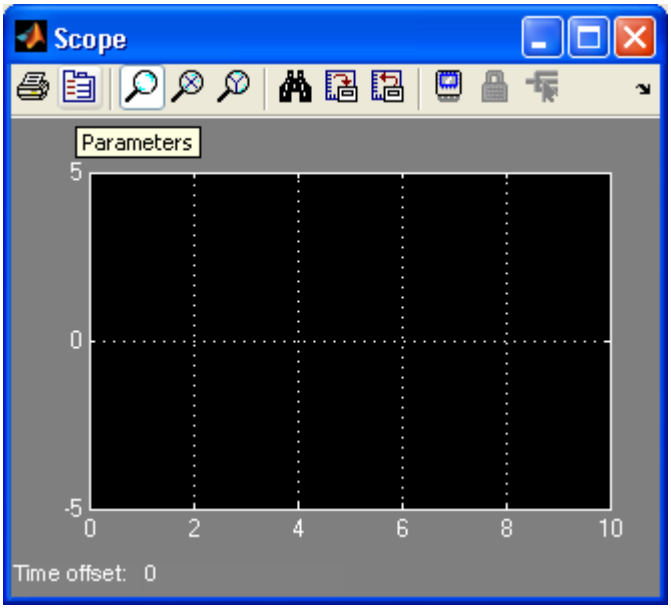
Εικόνα 6.28

Στην επιλογή *Solver Options-Type* επιλέγω *Fixed-step*. Στην επιλογή *Solver* επιλέγω *ode5 (Dormand-Prince)*. Στην επιλογή *Fixed step size* box βάζω ένα χρόνο 0.001 second. Στην επιλογή *Tasking mode for periodic sample times* box επιλέγω *Single Tasking* (Εικόνα 6.29).

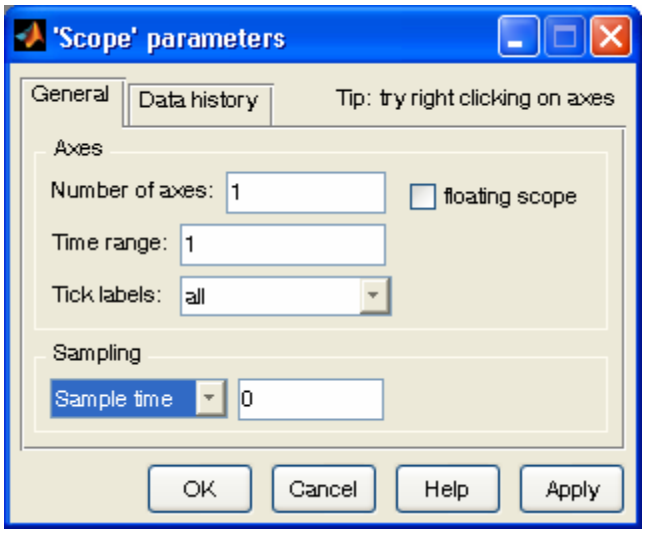


Εικόνα 6.29

Στην συνέχεια κάνω διπλό κλικ στο μπλοκ *Scope*. Κάνω κλικ στην επιλογή *Parameters* και βάζω τις παρακάτω παραμέτρους Εικόνα 6.30, Εικόνα 6.31).

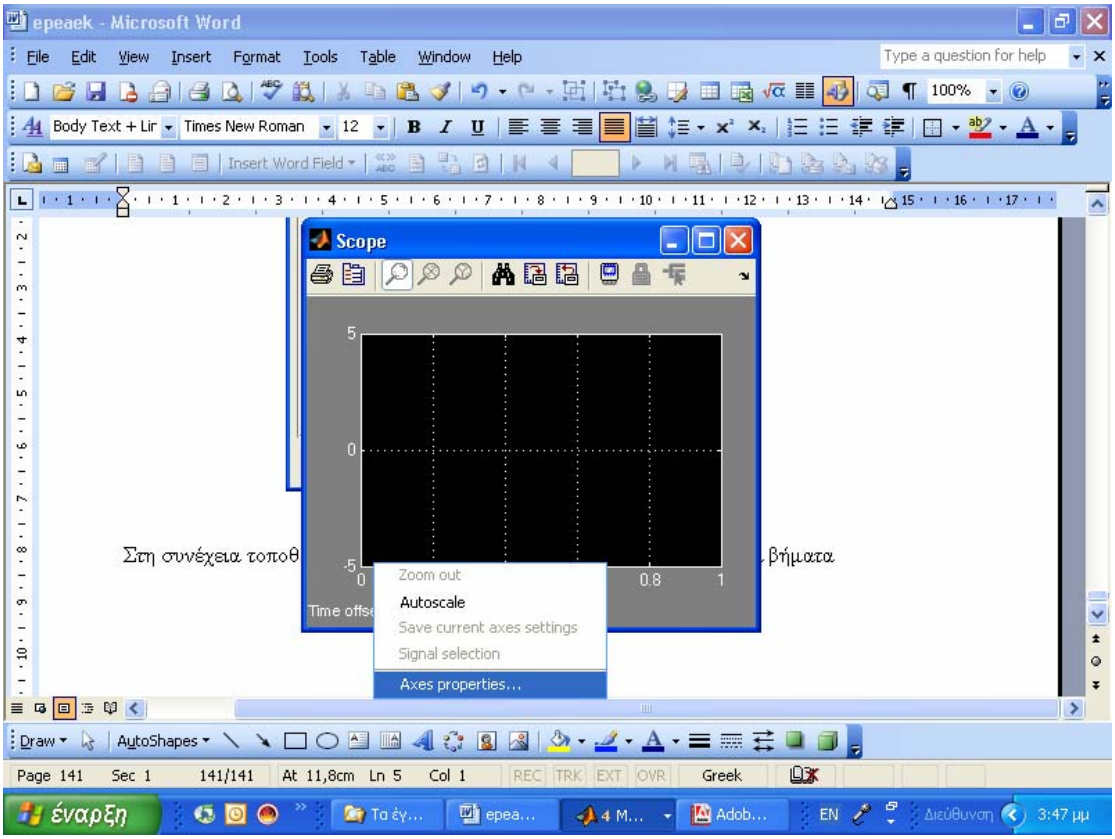


Εικόνα 6.30

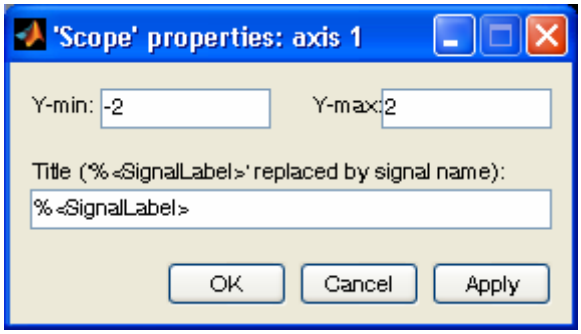


Εικόνα 6.31

Στη συνέχεια τοποθετώ τη κλίμακα πάνω στους άξονες με τα ακόλουθα βήματα (Εικόνα 6.32, Εικόνα 6.33)

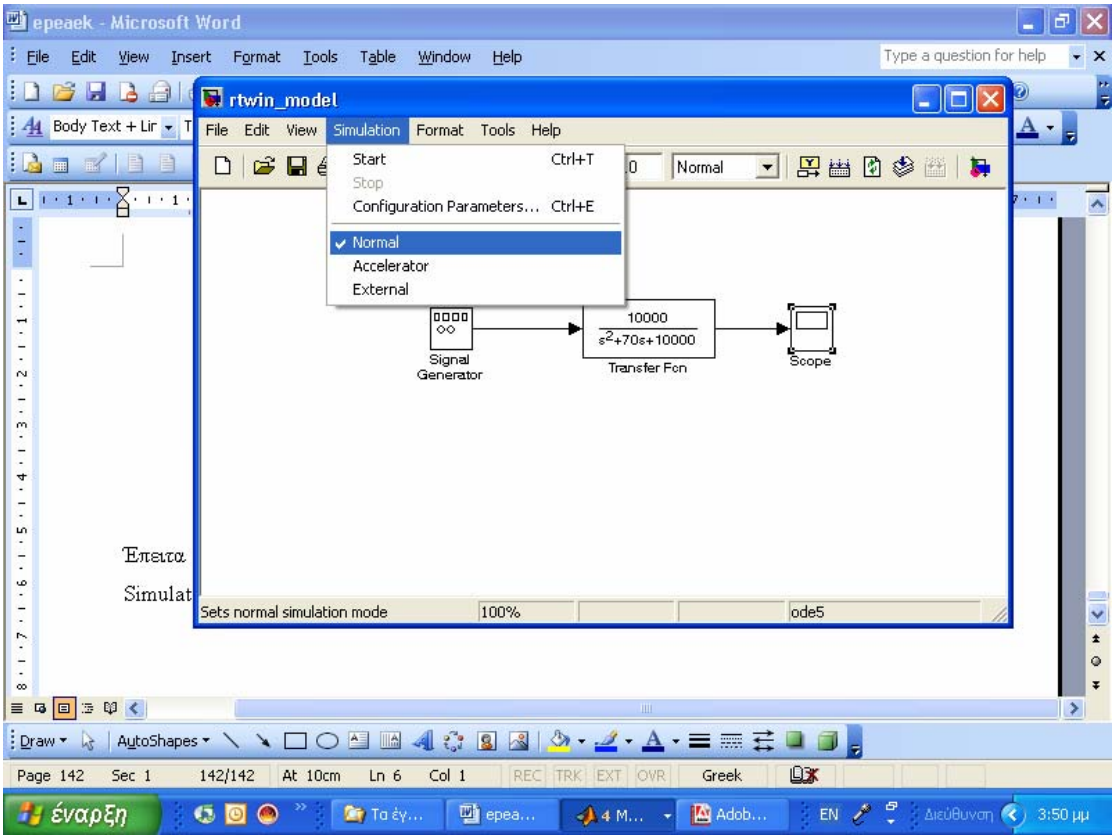


Εικόνα 6.32

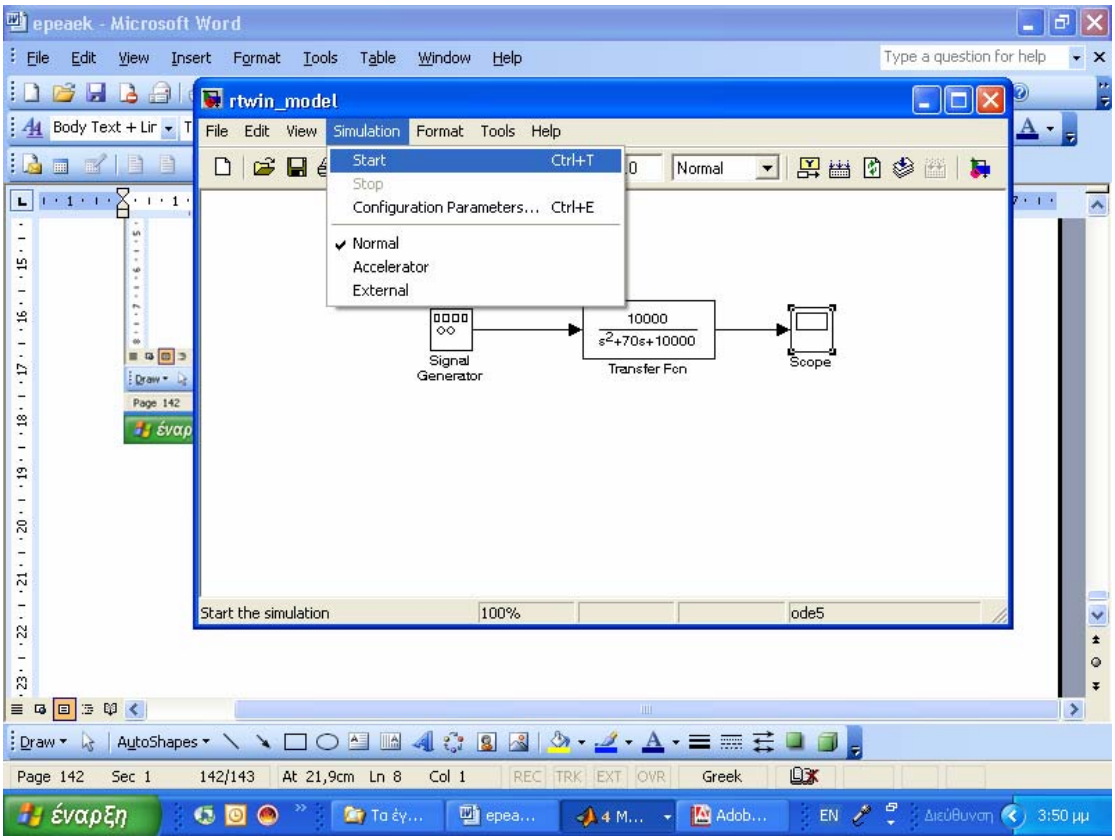


Εικόνα 6.33

Έπειτα «τρέχω» το μοντέλο επιλέγοντας *Simulation-Normal* και έπειτα επιλέγω *Simulation-Start* (Εικόνα 6.34, Εικόνα 6.35).

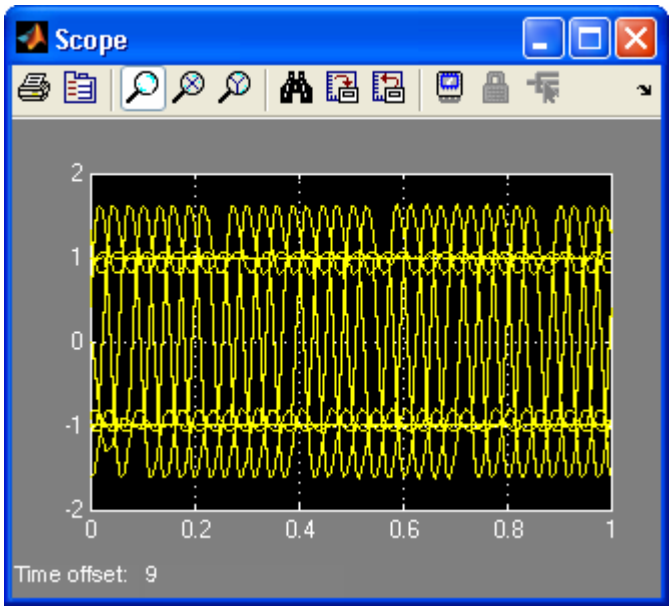


Εικόνα 6.34



Εικόνα 6.35

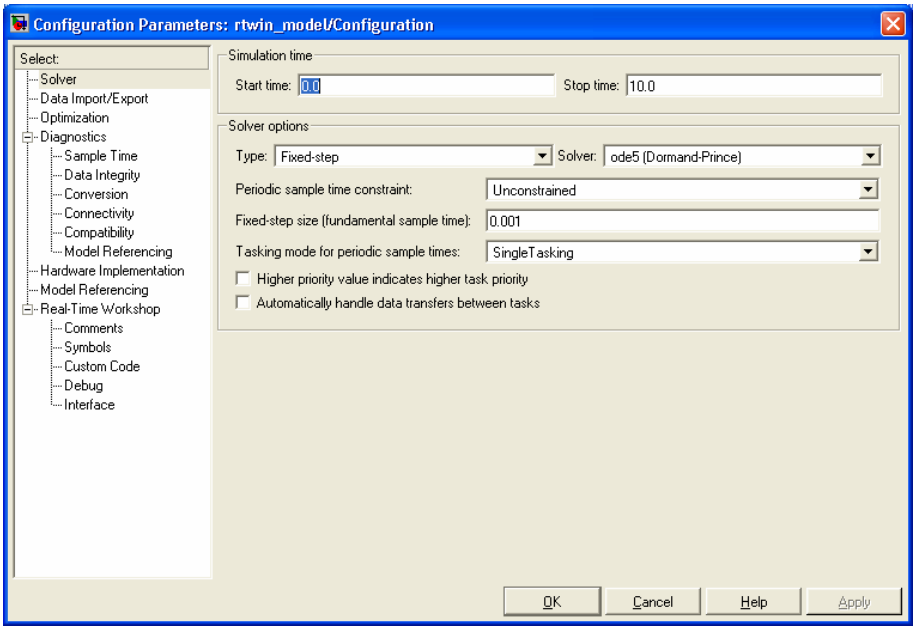
Το μπλοκ του *παλμογράφου (Scope)* δίνει τελικά στην έξοδο του συστήματος (Εικόνα 6.36)



Εικόνα 6.36

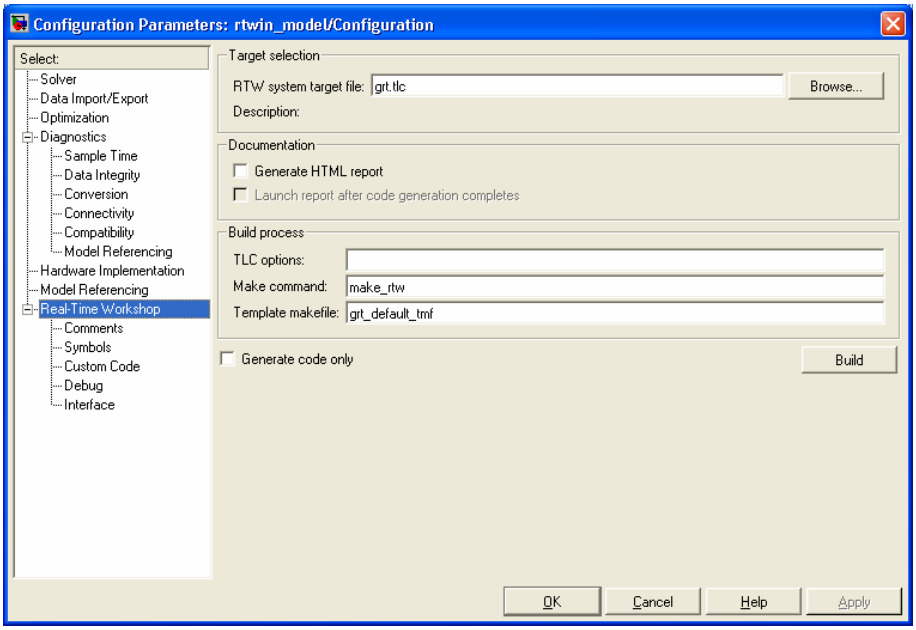
6.6. Κατασκευή μοντέλου πραγματικού χρόνου σε Simulink

Το μοντέλο που φτιάξαμε πριν θα το μετατρέψουμε σε μοντέλο πραγματικού χρόνου (Real-Time). Αρχικά αλλάζουμε τις παραμέτρους. Από το *Simulation* μενού κάνω κλικ στο *Configuration Parameters* (Εικόνα 6.37).



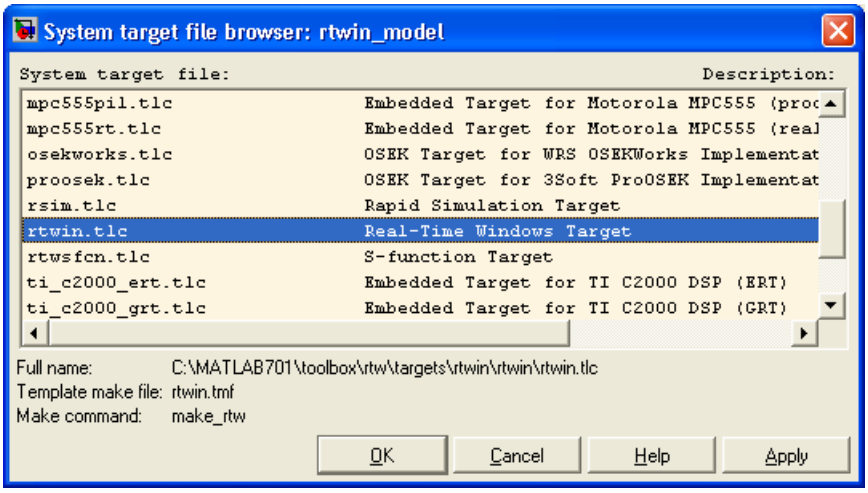
Εικόνα 6.37

Στη συνέχεια από τη αριστερή λίστα κάνω κλικ στην επιλογή *Real-Time Workshop* (Εικόνα 6.38).



Εικόνα 6.38

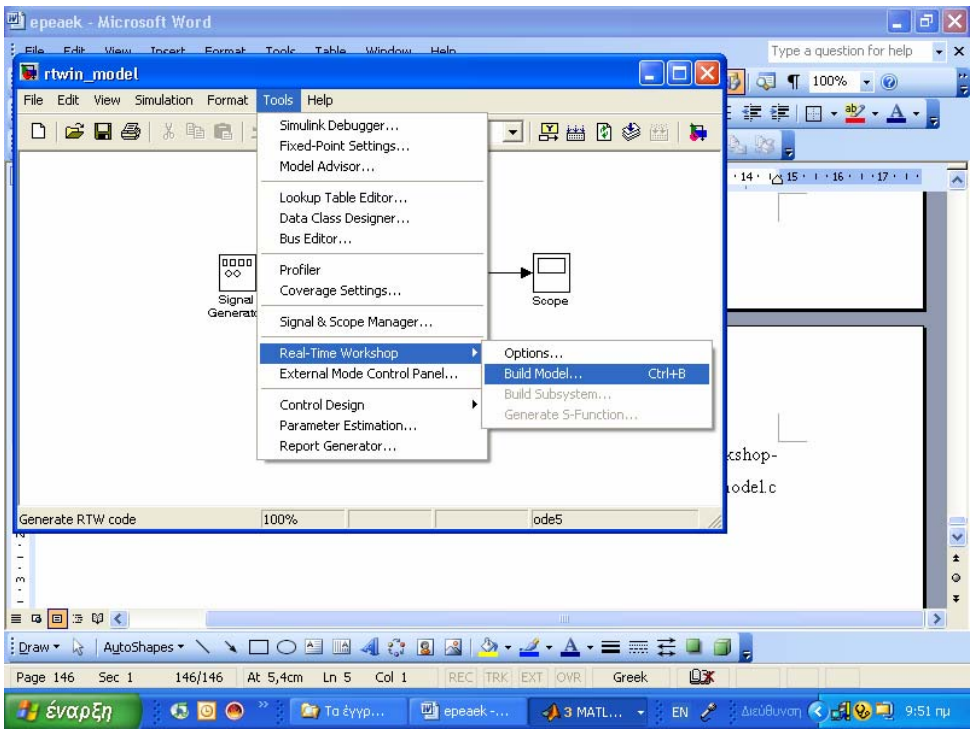
Στη θέση *Target selection* κάνω κλικ στο κουμπί *Browse*. Στη λίστα που εμφανίζεται επιλέγω το αρχείο συστήματος για *Real-Time Windows Target* εφαρμογές (Εικόνα 6.39).



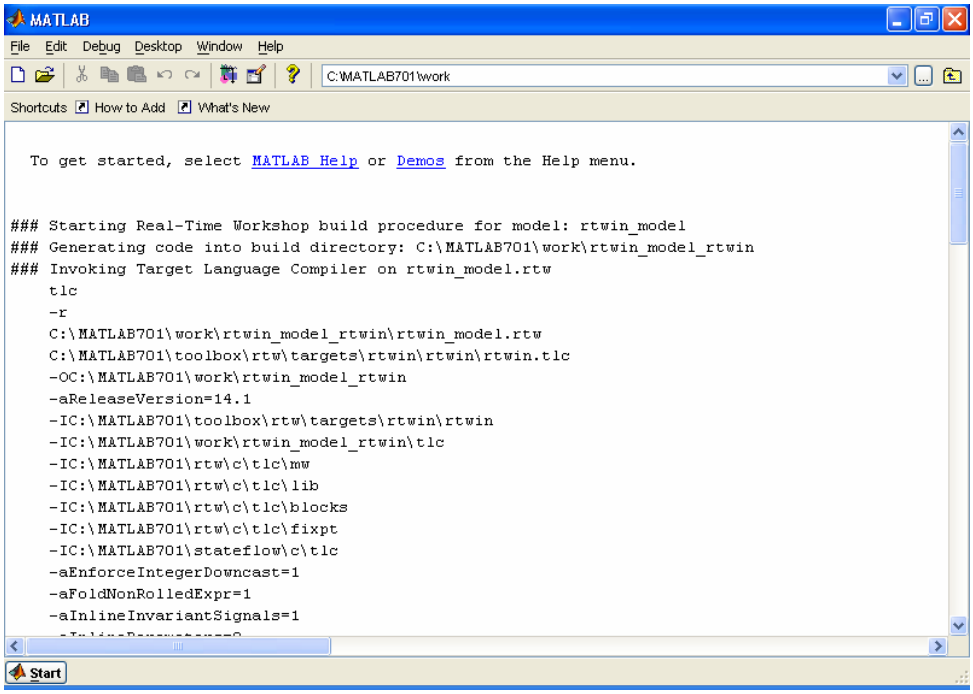
Εικόνα 6.39

Στη συνέχεια κλείνω το παράθυρο *Configuration Parameters*. Δεν μεταβάλλω τις παραμέτρους στα μπλοκ που έχω χρησιμοποιήσει.

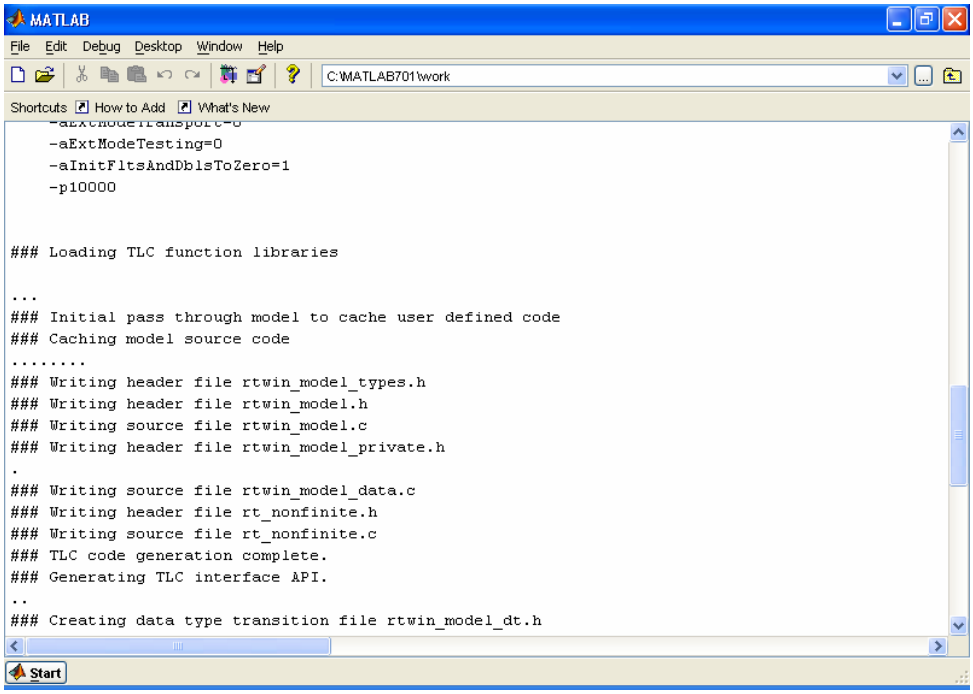
Για να «τρέξω» το μοντέλο επιλέγω από το μενού *Tools->Real-Time Workshop->Build Model*. Οπότε το *Real-Time Workshop* δημιουργεί τα αρχεία *C rtwin_model.c* και *rtwin_model.h* (Εικόνα 6.40, Εικόνα 6.41, Εικόνα 6.42, Εικόνα 6.43)



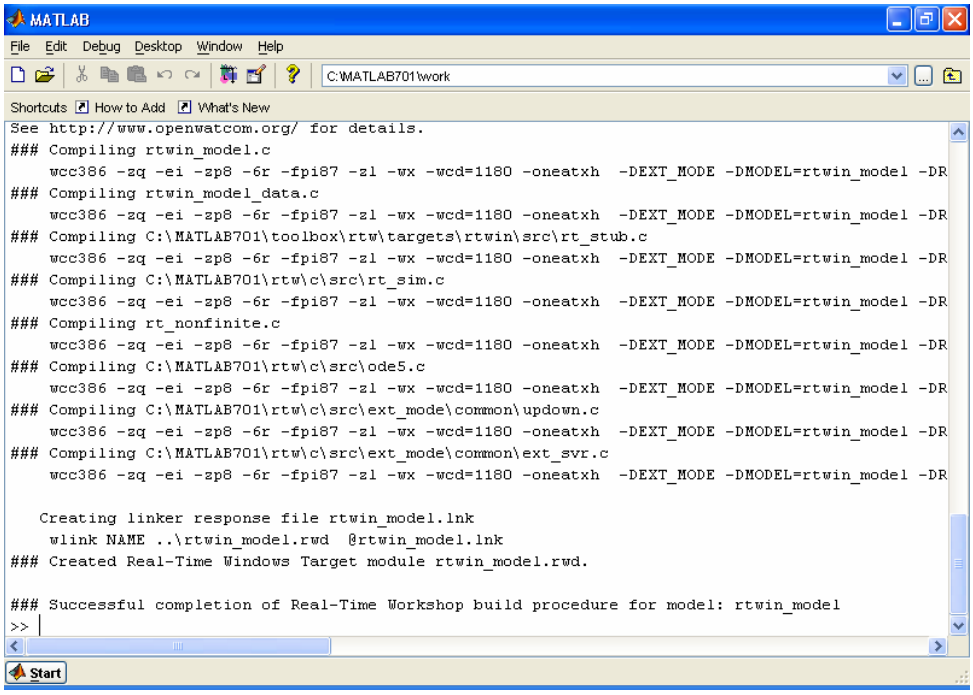
Εικόνα 6.40



Εικόνα 6.41

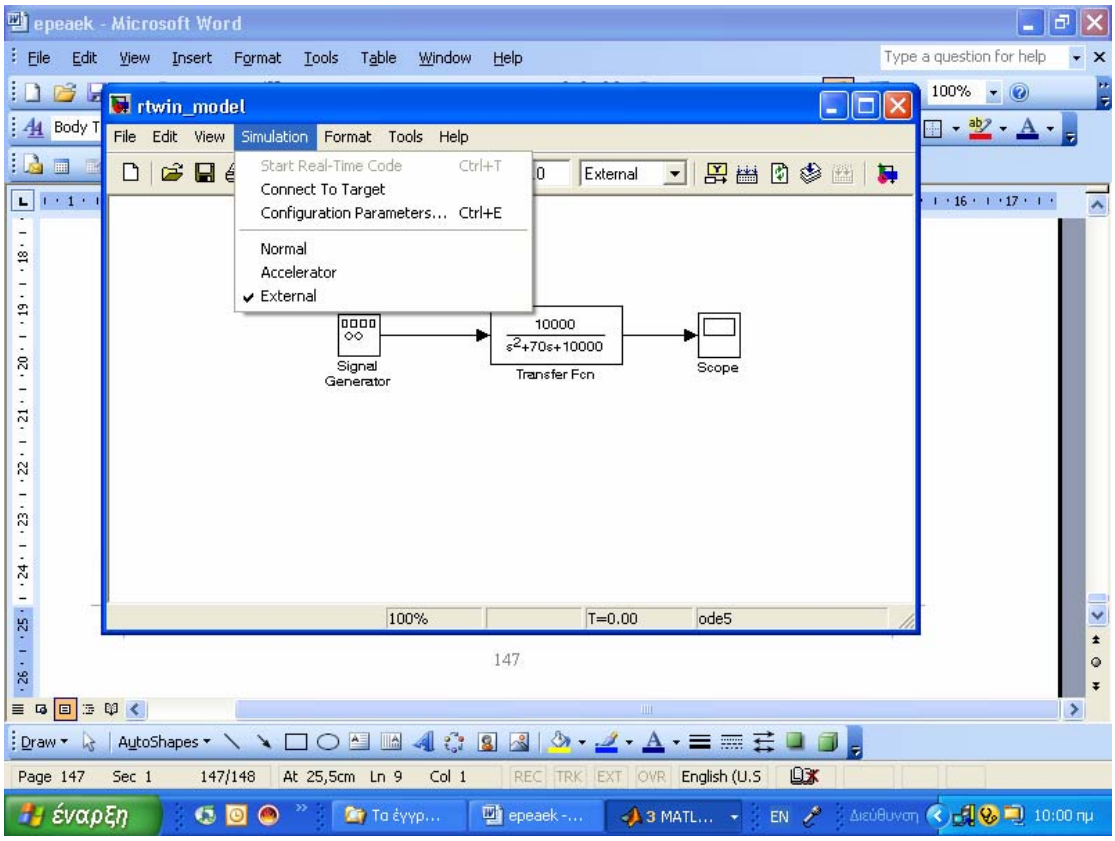


Εικόνα 6.42

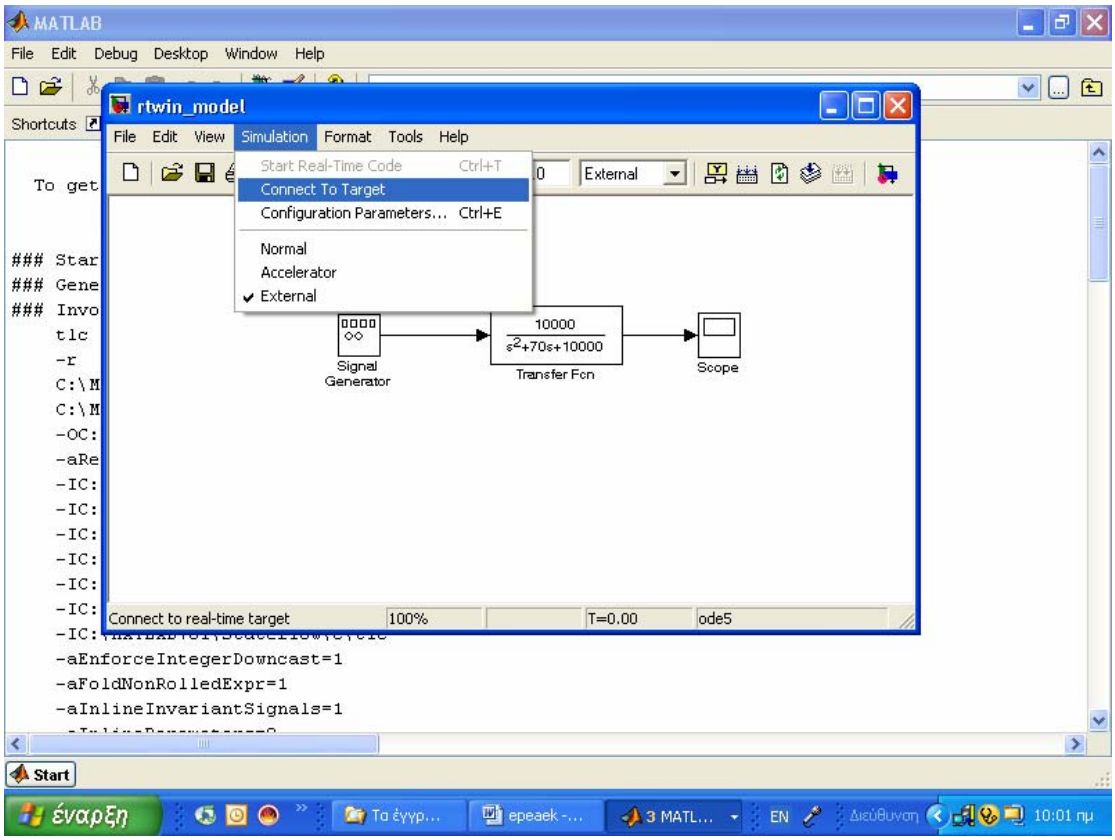


Εικόνα 6.43

Στη συνέχεια για να «τρέξουμε» το μοντέλο επιλέγουμε *Simulation-External* και στη συνέχεια επιλέγουμε *Connect to target* (Εικόνα 6.44, Εικόνα 6.45)

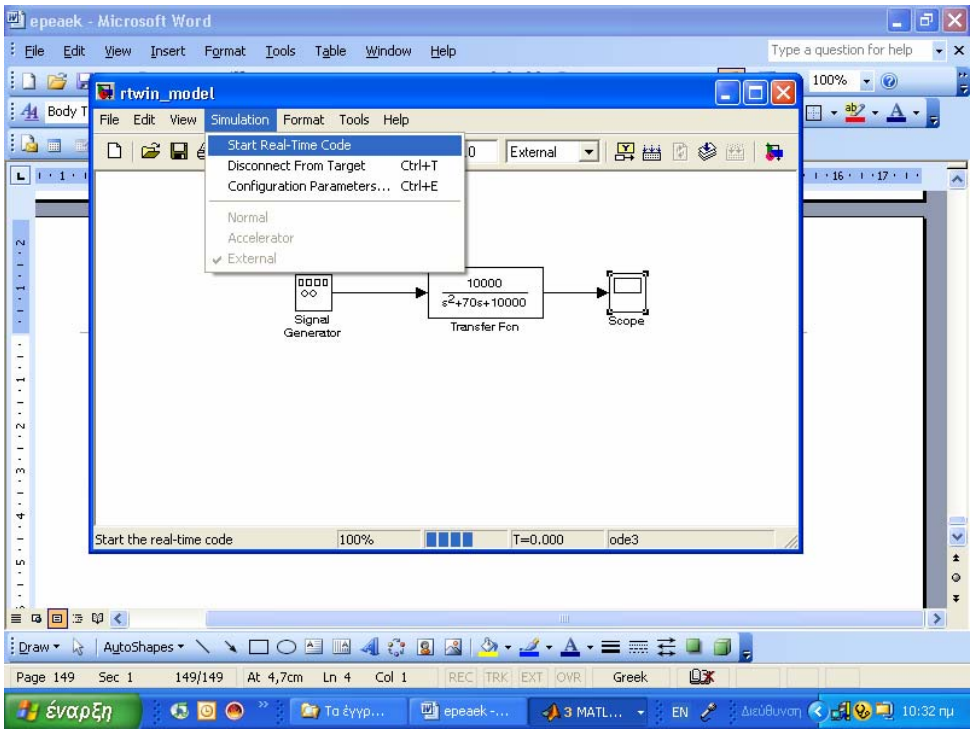


Εικόνα 6.44



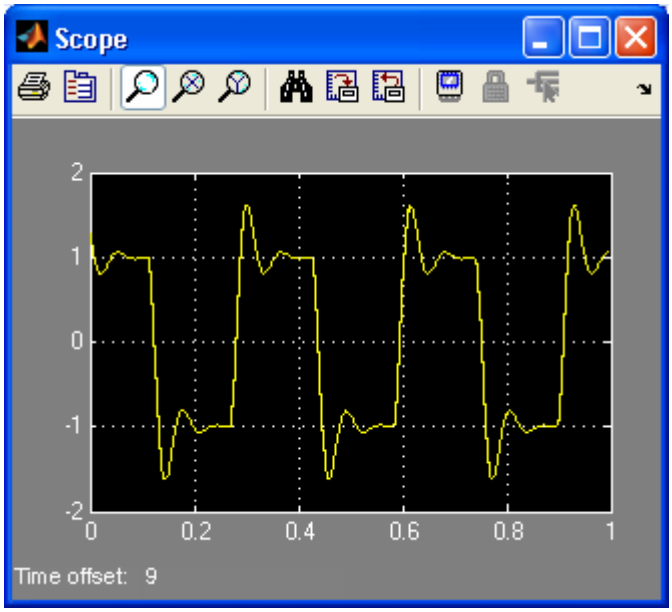
Εικόνα 6.45

Στη συνέχεια επιλέγω *Simulation-Start real-time code* και το μοντέλο «τρέχει» σε πραγματικό χρόνο (Εικόνα 6.46)



Εικόνα 6.46

Και το μπλοκ του *παλμογράφου (Scope)* δίνει (Εικόνα 6.47) στην έξοδο του συστήματος (εφόσον έχει ολοκληρωθεί ο χρόνος προσομοίωσης)



Εικόνα 6.47

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. MATLAB 6 Κωνσταντίνος Παπαρίζος
2. MATLAB 6.5 Κωνσταντίνος Παπαρίζος
3. ΣΥΝΟΠΤΙΚΟΣ ΟΔΗΓΟΣ MATLAB-SIMULINK Νεκτάριος Αρναουτάκης
4. Real-Time Windows Target for use with Real-Time Workshop User's Guide